ISSN 0112-3351

BEEBLET

CONTENTS

Editorial Notes 2
President's Report 3
HELP 7
Classified Ads 8
Electron Column 9
Memory Saving 11
Crossword 12
Hall of Fame 15
Mustave 16
Switched Joystick Interface 18
Electron Ram Board 22
The Graphics ROM - Review 23
Beginner's Seminar 25
Beta Based Files - Review 29
Don't CLS 31
Christmas with Strings 32
3D Text 34
Rehash 39
Finding Strings and basic Keywords 43
Getting the Most from Your BBC -Review 44
Handbook of Procedures and Functions 48
Memory Dump 49
The Perfect Programmer 52
Enlarged Printing 53
Space Quest 59
RTTY -Review(s) 64
Amateur Radio Software 67
A Printer? 71
Random Numbers 75
Program Recovery and Token Printout 77
Local Groups 80

ADVERTISERS

APTEC 78

VOLUME 3 NUMBER 11 DECEMBER 1985

BBC / ACORN

COMPUTER USER GROUP NZ (inc)

EDITORIAL NOTES

Seasons Greetings! I hope that the following few pages will fill in a fair few hours over the holiday break. Watch out for the crossword though - its not quite as straight forward as last years.... Incidently, THREE different Tony W.s contributed to this edition - and I suspect the crossword author may wish to preserve this relative anonymity!

COMPETITION

Gavin Flower's SAS100 and Bill Tillick's T-Draw and Sub Hunt have been won the software competition that I announced in April. Congratulations to both of you.
The contributions of Tim Ryan (Windows), Neil Gordon (Disk Backup) and Alan Beeston (Printing Disc Maps) were also very highly commended.

A detailed account of the criteria used in judging will be supplied for the February issue by Warren Willis who very kindly agreed to do the judging.

THANKS

I should like to thank all those people who have put in a great deal of time and effort this year assisting me in magazine production - Anne Flower in particular, and Gavin, Tim Ryan, Steve Cosgrove. Thanks also to those who have contributed material - a lot of people all round the country have put in very coniderable energies to produce generally quite a high standard of contributions. Please continue this next year...

APOLOGIES

....to those such as Tony Watton (Beebs in the Backwoods) whose names I have omitted from their contributions (I suggest contributors embed their names firmly into their work), and to all those I have not yet personally written to thank for their efforts.

FINALLY..

Please enjoy the break - I certainly will - but remember to send in something for the February edition. Don't be surprised if correspondence is somewhat delayed over the next few weeks either. You might start thinking about who might be willing and able to take over Beeblet next year - the current Editor is due for the 100,000 mile overhaul.......

****

BEEBLET is edited by Bruce Wills who may sometimes be tracked down on 861-662 (home), or 729-929 (work).

DISCLAIMER

The views of the reviewers, the Editor and other contributors are not necessarily shared by the
BBC/ACORN COMPUTER USER GROUP NZ INC

PRESIDENT'S REPORT - DECEMBER 1985

MODEMS

As I write, the decision of the Post Office on our modem has yet to be officially notified. Unofficially, it seems to have passed with the exception of the 12V AC power pack we used, which apparently didn't have a NZ Standards certificate. An alternative is being provided to them.

At an expected price of around $210, the modem is an incredibly good deal. The Fountain modem, with comparable facilities, is now on sale in Wellington for $399 plus another $30 or so for the cable to connect it to a BBC. If you haven't already done so, send in your $50 deposit to secure your modem (see last months "Beeblet" for details). Delivery will now, obviously, be early next year.

QUESTIONNAIRE RESULTS

Thanks to all those people who took the time to complete and send in their questionnaires. In particular, I thank those who added comments - these will be taken into account in our endeavours to continue having the best possible User Group.

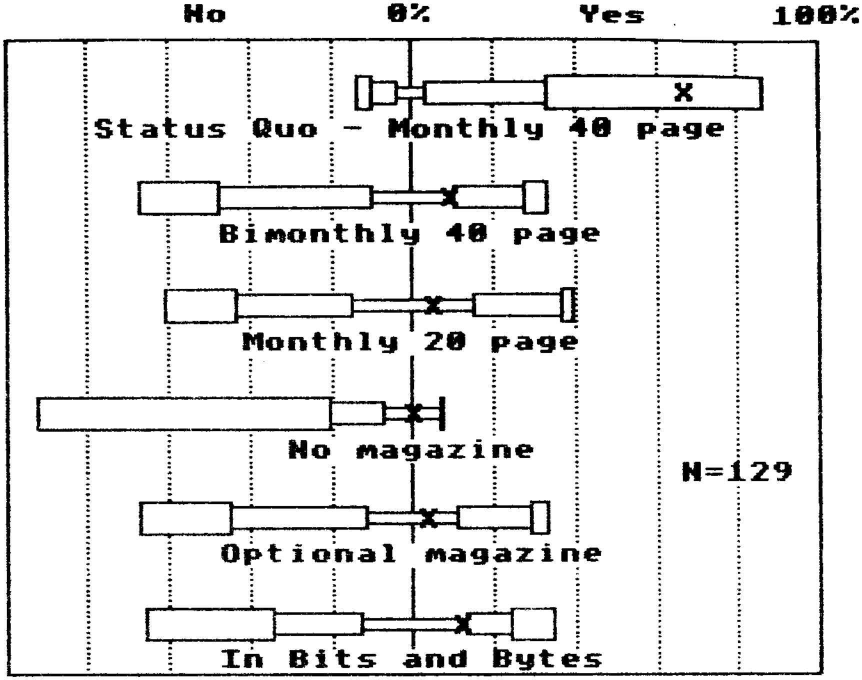
As the October magazine was published a little later than expected, we extended the deadline for replies to the end of November. By then, 129 replies had been received from financial members.

The questionnaire was also sent out to non-renewing members. Two of them replied and rejoined! They have been counted as financial members. Five others also returned questionnaires. One no longer had a BBC and didn't actually answer any of the questions. Another thought a one page local monthly newsletter was all that was required, and might rejoin if there was either no magazine or it was optional. The other three were generally in favour of the magazine (either monthly or bimonthly).

Magazine Options

The diagram below summarises the responses for the six magazine options. It's quite simple. Really.

For each option, the length of the central (thinnest) bar represents the percentage of people who either ticked "Don't Know" or, presumably not knowing whether they didn't know, didn't tick any box for that question. This bar is centred on the zero line.



To the left, the length of the thicker bar indicates the percentage of people ticking "No", and the thickest bar the percentage ticking "Definite NO". Similarly, on the right the thicker bar represents those ticking "Yes" and the thickest bar those ticking "Definite YES".

So, if we allocate half of the fence sitters to each side, the total length on the left indicates those people not in favour of that option and the length on the right those who like it. Adding up those two lengths gives 100%.

In addition, the X shown in the Yes area for each option indicates the proportion of people who gave that option their highest rating of the six options. (If two or more options were equally highly rated, the "votes" were split between them.)

The results resoundingly favour the status quo - about 85% were generally in favour of a monthly 40 page magazine and 68% chose it as their most favoured option. Only a few percent sat on the fence for this one. The bimonthly or monthly 20 page options were much less favoured.

It is also clear that almost everyone wants a magazine of some sort. Only 0.6% chose "No Magazine" as their most favoured option, and 4% "Optional Magazine". While just over 30% were generally in favour of an optional magazine, in the subsequent question about whether they would want "Beeblet" if it were optional, only 12% said they wouldn't get it. 82% said they would get it, and 6% gave no answer.

The questions about inclusion in Bits and Bytes" led to some interesting comments - ranging from "what is Bits and Bytes?" to "it's terrible and I won't be renewing my subscription" to "everyone gets it anyway so that's a good idea". This option came second in the popularity stakes, with 13% giving it their highest rating. However, the result is somewhat academic as, judging by their lack of reply to my letter, Bits and Bytes now don't seem interested anyway.

Finances

Judging by the comments on this question, it seems to have been misinterpreted by some. I apologize if I did not make it clear that we were looking for a way to return some of our accumulated surplus to loyal members, rather than curing any problem of tardy renewals.

The most popular choice for a rebate was $5, with 56% support. 36% wanted no rebate at all, and only 6% wanted a $10 rebate. Two people didn't know. The general impression from the comments was that the accumulated surplus wasn't very large and should be put to good use in improving services for the membership.

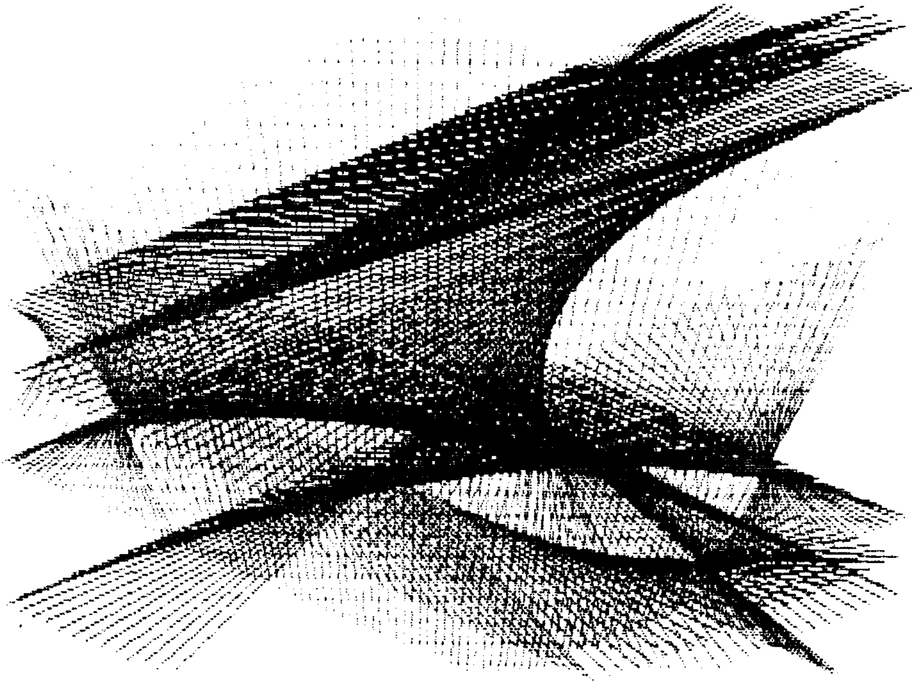
Summary

It is clear both from the above statistics and comments written on the bottom of the questionnaires that the magazine is seen as essential to the User Group. Many people commented that "Beeblet" is their only contact with the Group, as they are not able to attend any User Group meetings. We will continue to produce "Beeblet" with its current size and frequency. Looking further in the future we may eventually move to some sort of Bulletin Board service containing all or part of the magazine, as suggested in a number of the comments.

Given the majority support for a $5 rebate for prompt renewals (within a month being suggested by many), this will probably be recommended to the AGM - subject to our financial situation at the time.

Finally, may I take this opportunity to wish you all a very happy Christmas and the best of things for 1986.

Neil Gordon



SIDEWAYS RAM

Anne F.

A member in Nelson has recently got the Watford MK 2 ROM/RAM board, which comes with fitting instructions, but no help on how to use the RAM that is fitted. Could other members who have some experience in this area please write in to help this member. Replies could be published for other members perusal also.

HELP!

A forum conducted by Tim Ryan

@%

Q1 Peter Till asks about a problem he has with @% not always rounding off and printing numbers like 243.33000001 when @% is set to &20206.

A1 I really can't give you a final answer on this one Peter. Either you've reset @% accidently - perhaps by overwriting memory location &400, or there is a bug in BBC/ELECTRON BASIC.

OSCLI and File Names

Q2 J Logie asks about using OSCLI to handle variable file names in BASIC - he found the answer in April's Beeblet (which wasn't mine) a bit brief and cryptic.

A2 OSCLI is the OS's way of handling any commands from the user (star commands).

OSCLI - being OS based - can not handle BASIC variables, so if the user wishes to pass variables from BASIC through the OSCLI vector to OS, the contents of the variables must be passed, not the variable name.

BASIC 2 can handle this easily, with the use of the reserved word OSCLI, BASIC 1 is a bit more complicated and will be explained in a moment.

In either case the command message should be set up in a string variable (eg- comm$) and called by:

BASIC 1 - PROCoscli(comm$)
BASIC 2 - OSCLI(comm$)

The procedure required for BASIC 1 is as follows:

```
DIM line 255:REM put this at the start of your program
DEF PROCoscli($line)
LOCAL X%,Y%
X%=line MOD 256:Y%=line DIV 256
CALL &FFF7
```

```
ENDPROC
```

Some example s of possible command lines:

```
comm$="DIR "+dir$
          where dir$="A"

result: *DIR A


comm$="SAVE :2.A."+filename$+" 3000 7C00"
          where filename$="SCREEN"

result: *SAVE :2.A.SCREEN 3000 7C00


comm$="LOAD :"+drive$+"."+dir$+"."+filename$+" "+str$~start%+"
"+str$~end%
          where    drive$="1",    dir$="B",    filename$="TEST",
start%=6400, and end%=32767

result: *LOAD :1.B.TEST E00 7FFF
```

CLASSIFIED ADVERTISEMENTS

For Sale:

BBC micro, issue 4 P.C. Board (2 years old). 1.2 OA, DOS, Wordwise and Star Data Base ROMS installed. Mitsubishi 80 track Double Sided Disc Drive (1 year old) utility disc and Disc User Manual. Games and Primary Education Software. $2,200 ONO.

Phone Evan 330-444 Invercargill

BBC B with DFS and lots of software : $1000. 400K Mitsubishi disc drive with utility disc and manual : $400-00.

Phone Tony Booth 639-155

electron column

steve cosgrove

No-one has offered to buy me a Plus 3 yet, so I have not yet written a review of it. (Wouldn't it be fun just to try offering one to me? - who knows, I might turn it down!) Notwithstanding that, Whitcoulls have lent me one for various research purposes occasionally, so I must put in a good word for the unit.

The compact nature of the Electron system is one thing that really appeals to me. Even with the Plus 1 & 3 all bolted together, I can still put it into a shoulder bag, with lots of padding, and carry it easily. The Plus 1 expansion sockets have all the pins which I will ever need to expand my system, so there will be none of the infinite tangle of wires, and extra boards which fill some BBCs. (Ed.s query: You are not getting at someone are you ? - so what if it works better with the top off..)

For those lesser mortals who haven't caught up with these "extras" here is a summary of their operations.

Plus 1 : Allows connection of a printer, joysticks, and two extra expansion sockets. In order to make all this work, an extra ROM (2K I think) is provided inside. This ROM also supports an 'RS423' addition, which is a serial interface. Basically, that is a means of communicating with other computers, and similar functions.

Plus 2 : Econet interface. Used mainly in schools. Any a number of computers, connected together by the Econet Network, can read programs from one computer (called the "server") with a disk drive. Not only is it convenient to be able to hold all the programs centrally this way, but it is also cheaper than have a disk drive on each computer. One printer can also be shared by all computers on the network. Resource sharing is

↓↓

just one of many features of Econet.

Plus 3 : Disk drive. No doubt most Electron users have seen a Plus 3 in action. On first inspection, one could be excused for wondering how an L shaped hunk of plastic can cost $1000.00 or so. A large amount of the cost is in the drive part. That is the bit that sticks out the right hand side. It is called a 3 1/2 inch drive (not even the Japanese use metric for that) because the "disks" which it uses are 3 1/2 inches in diameter. The disk is held in a solid plastic case, with a shutter over the only opening, which is where the drive gets to read data stored on the disk. This type of disk looks to be becomming the new "standard", as it is a convenient size, and in not very susceptible to damage. It can hold the same amount of data as the larger (5 1/4") disks that have been "standard" for some years. At present 3 1/2" disks are more expensive, but as their use increases, the price will decrease. (Current User Group prices are $55.00 & $85.00 respectivly for boxes of ten.)

The real beauty of the Plus 3 is it's "ADFS", which is the way in which it allows the user to save programs on a disk. You probably have tried to find a document in an office where everything is filed on one place. It takes ages. A disk can be the same if a lot of programs are saved on it. However, the Advanced Disk Filing System allows creation of any number of separate "directories".. A directory is like a folder or drawer in a filing cabinet. A number of sub-directories can be made up for different categories of programs, and relevant programs stored therein. Sub directories can be further divided, if need be. For example, I have a disk which has directoies called "basic" "Pascal" "newsletter" for different types of program. "newsletter" is further devided between "current" and "old". At the end of each month, I copy everything from the "current" directory to the "old" one.

Another thing I have discovered this weekend, (indeed, the reason I had been loaned it) is that it is very easy to connect a second disk drive to the Plus 3, which makes copying disks considerably faster. The drive we used is 5 1/4" size, which allows reading of different sized disks. It could easily have been another 3 1/2" one.

Thats it for this month. Have a good time over Christmas. As always, letters on any subject are welcome. I do try to reply quickly to those which I get. Please note too that we are developing a RAM board for the Electron. A notice about this appears elsewhere in the newsletter.

↑↑

MEMORY SAVING

Steve Cosgrove

Well! It has arrived! The great Acornsoft Pascal in an Electron cartrige! Along with it came the "Pseudo Mode 7". This program uses the same technique to allow more memory for your program.

In my column last month, I talked about the extra memory demands brought about by not having Mode 7. If a program does not make much use of the screen (eg, only has to ask questions of the user), then a screen size of 6 lines by 40 characters could be all you need.

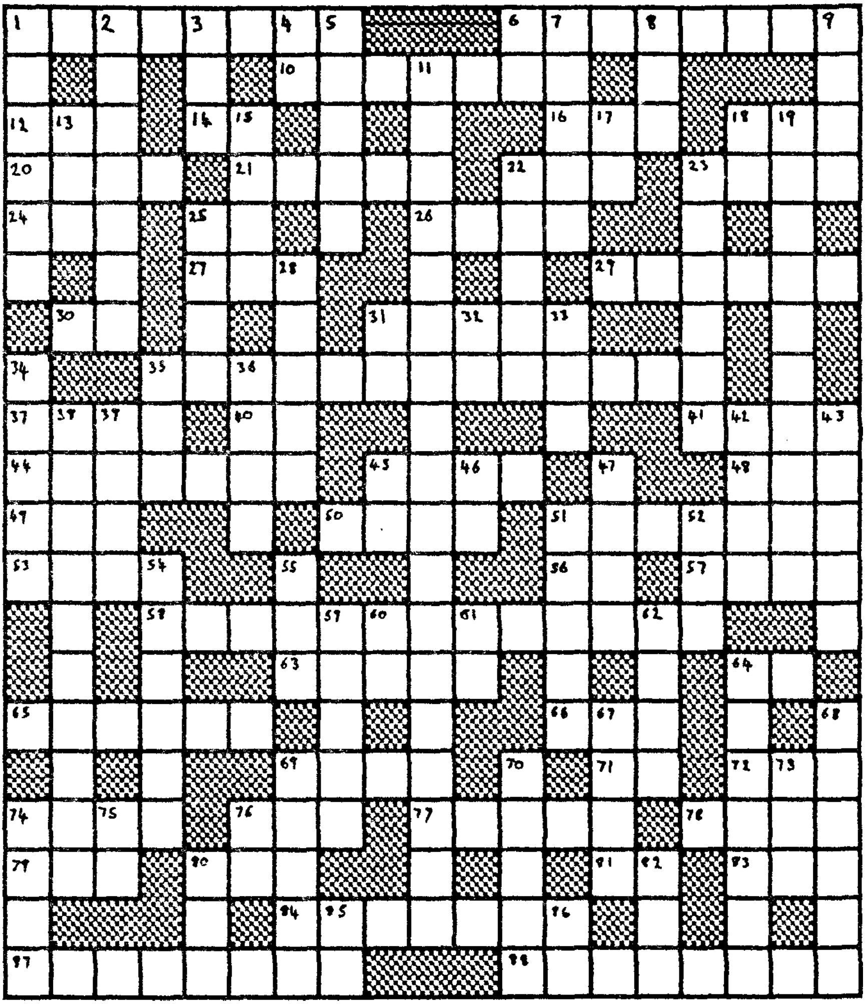
The program featured below sets a "window" on the screen, of that size, then resets "HIMEM", which is a marker used by Basic to work out how much memory it can use. Normally HIMEM is immediately below the screen memory.

Chain this program, then one which previously created a "no room" error. Before you run the second program however, make sure that it does not have any mode changes, as these will now generate an error.

```
 10 REM Extending available memory
 20 REM by Steve Cosgrove
 30 REM 8th December
 40
 50 MODE 6
 60 VDU 28,0,24,39,18 : REM text
 70 REM VDU 24,0;0;1279;256; :REM graphics
 80 HIMEM = &7600
 90 delay = 200 :REM hundreds of a sec
100 REPEAT
110   time% = TIME
120   REPEAT
130     PRINT "this is all the screen space ";
140     UNTIL time% + delay < TIME
150   time% = TIME
160   REPEAT
170     GCOL RND(4),RND(16)
180     PLOT 85, RND(1079),RND(256)
190     UNTIL time% + delay < TIME
200   UNTIL FALSE
```

CROSSWORD

Tony W.



ACROSS
1 Vigorous sport sort that lice reorganised (8)
6 Having the gall to misspell will create a star (8)
10 Party attendance payment (7)
12 They've one area of NZ computer media all taped up! (3)
14 Not you (2)
16 Sea bird of Santa's clime (3)
18 You haven't until it's all filled in (3)
20 Give backwards (4)
21 Easily copied disks? (5)
22 This useful hint may upset you (3)
23 Disassociate matter (4)
24 Basis of Santa's homeland (3)
25 Forward foreword of pirates cry (2)
26 Moderate collection of abodes (4)
27 A period sounding like program problem (3)
29 Santa's sleigh could use one when it's snowing (6)
30 We fifty odd territorys (2)
31 Cycle forerunning an execution (5)
35 Heart of lesser beasts (5,4,1,3)
37 A new thought - seeing deer for the blind ?? (4)
40 Numero Uno (2)
41 Low level plant operating system (4)
44 Break before we restart again (3,4)
45 The person or thing just mentioned (4)
48 Pod encapsuled dimension of a brain (3)
49 Single pointed card (3)
50 Pixels for the layman (4)
51 Tangential separation of a sector (7)
53 Fail to win twisted sole (4)
56 What a laugh (2)
57 Against mixed-up Tina (4)
58 Pious vaction (6,7)
63 For eating Xmas pud (5)
64 Asciilian return (2)
65 Register your social standing (6)
66 Program terminator (3)
69 More inversion (4)
71 Meaningless aboutface of 25 across (2)
72 Overflowed into stable about this time (3)
74 Round of visits (4)
76 Counter for serving stiff rod (3)
77 Latest Naval ranking co. beasty (5)
78 What Santa might collect on the way 64 down (4)
79 Contracted data entry (3)
80 Foxy place for computer to reside (3)
81 One third of seasonal greeting (2)
83 Nothing pointing to nowhere (Pascal) (3)
84 Bees not Beebs provide rods of enlightenment (7)
87 Santa's dear helpers (8)
88 A stratum below (3,5)

DOWN

1 Santa's home (6)
2 Real computer people (7)
3 Tree 'em for Kay 8 down ? (3)
4 Reciprical circumference over too are ? (2)
5 Ten million rupees to restate Corer! (5)
6 Sounds like a previous 54 down (6)
7 A bad mark on 43 down rearranged (5)
8 A dear small relation of Beeb (3)
9 Nice sort (4)
11 The heart of real beasties (5,5,8)
13 Reverse analogue to digital converter (3)
15 Formery or alternative to an exclusive alternative (4)
17 Negated down (2)
18 Us sounds tiny (2)
19 What we three kings brought each (3,7)
22 Binary times (5)
23 It's a bit dicy trying to figure out this generation (6)
25 Abominable creature (4)
28 Le rat changed (5)
31 Asciilian form feed (2)
32 Monitor alternative (2)
33 Really cooking (3)
34 The decider and the last! (5)
35 Utter express in two bytes (3)
36 Short form seasonal holiday (4)
38 For your war services and Xmas party (10)
39 Feminine memorys ?? (4)
42 Nope to unclosed (4)
43 Cloth jumbled 7 down (5)
45 Go less banished command (2)
46 Since (2)
47 57,005 say Hex is done for (4)
51 Polished programs attain this attribute (5)
52 Upturn a tropical tuber to obtain a luny period (3)
54 A person that's a program (6)
55 DFS in another world (3)
59 On a level above the sole (5)
60 Hither 45 down again! (2)
61 Assert one true high (2)
62 Tandy without a head (4)
64 No burgular alarms on Santa's entrance (7)
67 He traveled in an arc and landed a' amount before Xmas (4)
68 Above the heads of all at 87 across (6)
69 Piecing tool used of old by night to spread Christmas views (5)
70 Chew on metal and swallow data (5)
73 Packet switching network users need identity (3)
74 A weeping pile of rows ?? (4)
75 A higher position for a small dog without a pea (2)
76 Half a Beeb (2)
80 Useless computer (3)
82 Even this won't make your programs run smoother (3)
85 Contracted accumulating registers ? (2)
86 Would take me to court but for the neglecte'e (2)

HALL OF FAME

PUBLISHER:	GAME	HISCORE	SCORER
Aardvark	Frak	1,026,820	Andrew P.
	Monsters	175,110	Geof P.
	Zalaga	590,100	Allister J.
		52,090*	Nick C.
Acorn	Arcadians	34,740	Rasik R.
	Aviator	5,610	Joanne C's Dad
	Countdown	246	Andrew
	Meteors	46,200	Andrew P.
	Monsters	149,730	Rupert T.
	Planetoid	413,875	Peter J.
	Rocket Raid	140,777	Mark R.
	Snapper	250,000	Susan
	Starship Command	9,495	Robert N.
	Super Invader	59,250	Kiran R.
A&F	Cylon Attack	157,970	Aaron
	Chuckie Egg	319,450*	Greg C.
Alligat.	Bug Blaster	37,642	Steve W.
Amcom	Fortress	89,900	Andrew P.
Bugbyte	Twin Kingdom	385	A. Beeston
Comp.Con	Android Attack	236,195	Peter J.
Dr Soft	'747'	205	Steve W.
Firebird	Bird Strike	11,340	David G.
		14,980*	Patrick C.
FG Soft.	Mutant Defender	6,900	Steve W.
	Lords of Time	925/1000	Steve W/Kerry K.
	Snowball	900/1000	Steve W./Dianne
Micro. Pow	Castlequest	50	Jeremy B.
Prog Pow	Felix in Factory	13,150	Duncan's Flatmate
	Frenzy	915,468	Dianne T.
	Killer Gorilla	91,000	Susan
	Laser Command	109,225	P. Leadley
	Moon Raider	112,350	P.Leadley
Sup.Soft	Q*Bert	43,825	Peter J.
	Mr Wiz	21,050*	Chris M.
	Roadrunner	118,900	Duncan's Flatmate
?	Dr Who	35,240	Duncan's Flatmate

*Electron Version. Note: Is there life after Elite?

HALL OF FAME

MUSTAVE

A sort of music note processor

Tony W.

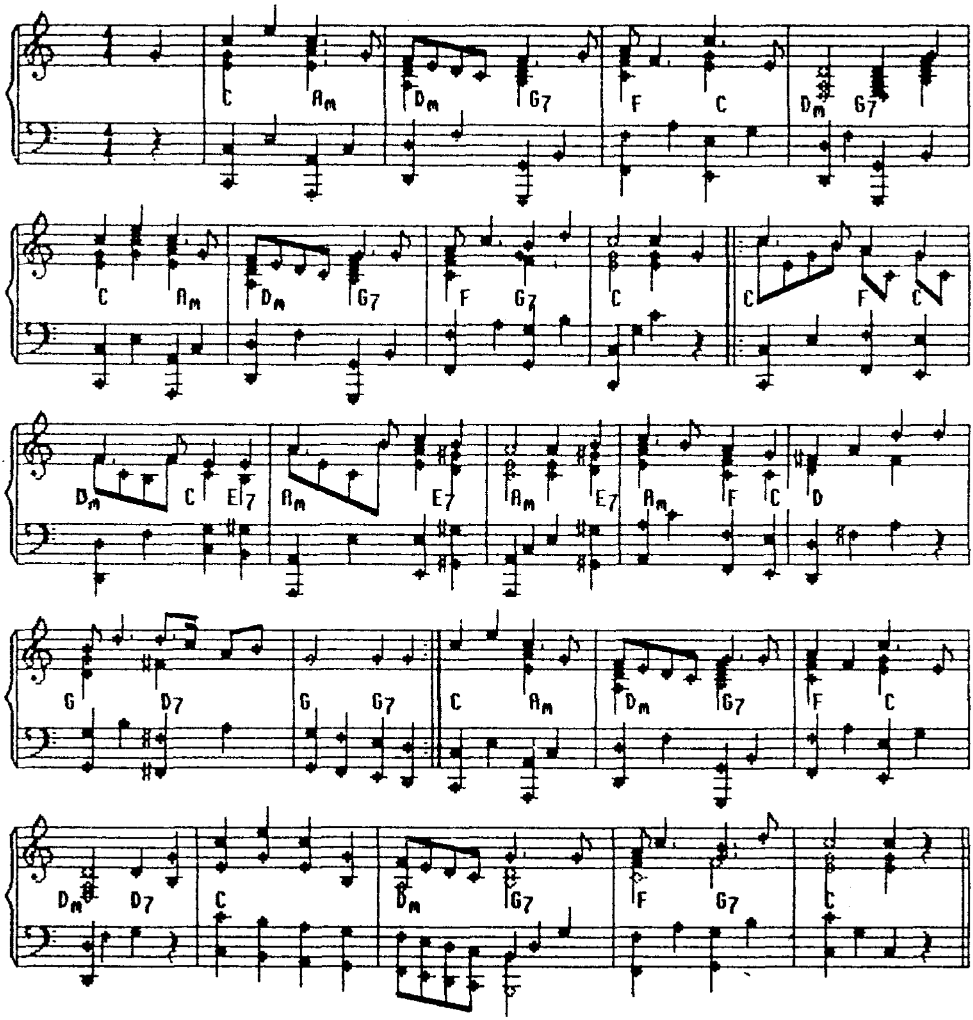
Editor's note: Tony recently delivered an interesting presentation, at rather short notice, to the Wellington group. While the program is currently unavailable, Tony has once again, at short notice, provided some details of some of the ideas he has been working on, and a sample printout. I hope this will stimulate a few other UG members with similar interests - Tony will be going overseas shortly, but would be interested in keeping in touch.

Programming in the W. household is a family affair. In the main I find myself having broad ideas but Julian is often called upon to add a bit of his knowledge and programming technique. I am not a graphic artist. Pieces of paper with writing, printing or music or any other information do NOT come from my hand. Therefore when, earlier this year, we keyed in a circuit draughting programme from a magazine, the idea was spawned for a kind of music note processor which would give a worthy print of music.

In its present form "Mustave" is two programmes, the first initiating the screen with a set of blank music staves while the second contains most of the user defined characters required in the printing of music. The joining of quaver tails has to be carried out with a pen and ruler on the printed sheet. (Not the work for a graphic artist). Depending on requirements, one, two or three staves will be produced on the screen. Other formats are planned, including those for songsheets and solo instrument with or without piano accompaniment. Saving is achieved by dumping the contents of the screen to disc. Likewise to print. Each screen dump on disc represents about 20K storage so that the capacity of an 80 track disc is only 9 files or screens. The programme has not been tried using tape mode but I suspect that loading and saving would be somewhat time consuming. The programme uses a machine code routine for dumping each screenful to printer. The printed block is 135mm wide full size which allows a generous margin on each side of a size A4 sheet. This programme was written to produce printed music scores with the advantages of edit, erase and copy features found on word processors. It has been used to produce the musical score of 11 tunes in a book shortly to be published. As an example, one of the tunes is reproduced here, by kind permission of the producers of the book.

A GUID NEW YEAR TO ONE AND A'

A GUID NEW YEAR TAE ANE AND A'



SWITCHED JOYSTICK INTERFACE

Paul Collins   Cambridge

Following pleadings from my son for "proper" joysticks and the aquisition of two Spectravideo Quickshot II joysticks at the right price, I set about building an interface that may be of interest to other members. The prototype was constructed on a piece of veroboard and has been very reliable.

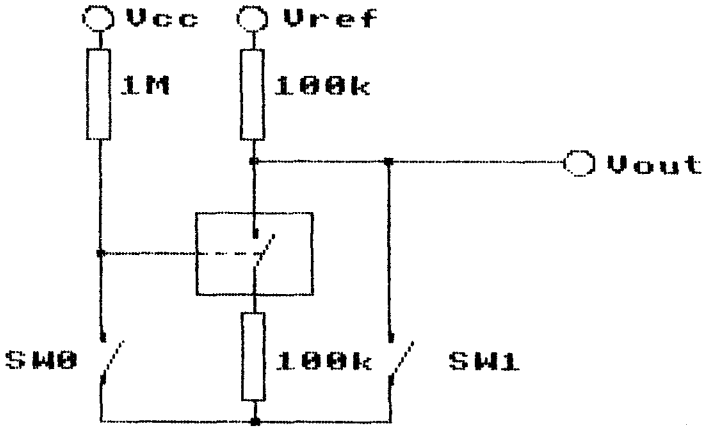
Please note the following:-

1. The interface is for joysticks switching to a common ground ONLY.
2. If only one joystick is required install all the R1 resistors to prevent parasitic oscillations in the CD4066.
3. VCC is only required if your joystick has 'AUTO FIRE' : CHECK CAREFULLY and if in doubt DON'T CONNECT.
4. The interface should be compatable with : ATARI, VIC20, C64 and SV 318/320 sticks but you pays your money and takes your chances.

As implemented here I have included a 15 pin socket on the box also, switching channel 0 to 4 and PB 0 to 1 between them with a six pole 2 way switch, the other lines being just paired as appropriate.

SPECTRAVIDEO QUICKSHOT II JOYSTICK INTERFACE FOR BBC MICROCOMPUTER

NB Should work with any joystick
   switching to a common ground.



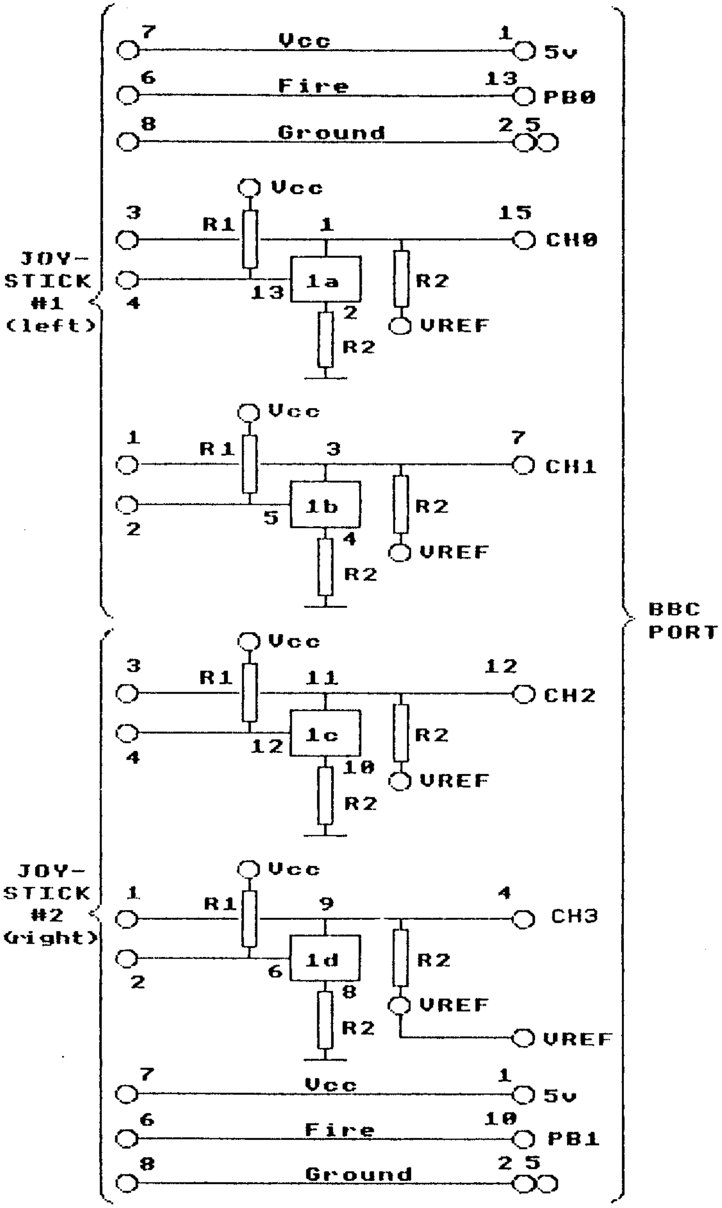
BASIC CIRCUIT DESIGN

SW0 & SW1 OPEN (centre) Vout=$^1/_2$Vref

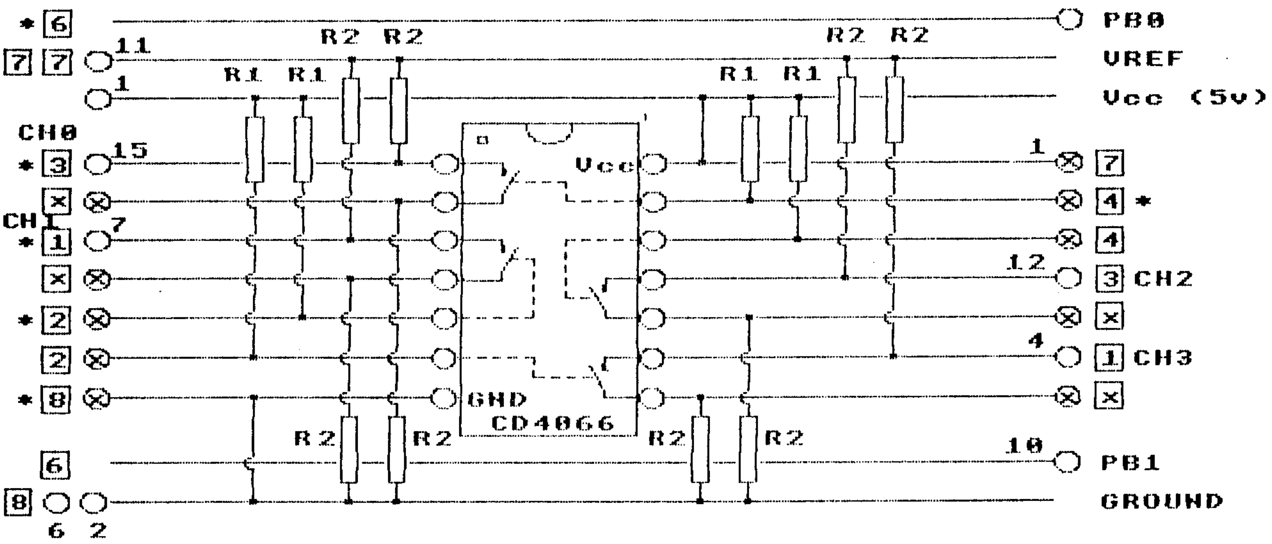
SW0 CLOSED Vout= Vref

SW1 CLOSED Vout= 0v

CAMCOM 22/9/85



R1 = 1MegOhm   R2 = 100kOhm
IC1 = CD4066 QUAD Switch



○ = BBC 15 Pin (x=no connection)

□ = JOYSTICK 9 Pin (*= #1)

R1 = 1Meg Ohm. R2 = 100k Ohm

Suggested veroboard for BBC switched joystick interface

ELECTRON RAM BOARD

Steve Cosgrove

Currently under development is an expansion board for the Plus One.
It will support any combinatoin of the following:-

32K of sideways RAM (with battery backup)

or

Two 16K BBC ROMS

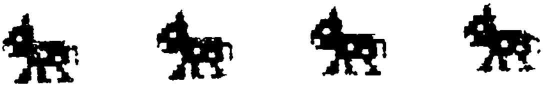
or

16K of sideways RAM and one 16K sideways ROM

This will allow Electron users to use ROMs such as LOGO, or Graphics.
A rough estimate of the cost of the board, with everything assembled, ready to plug in the RAM/ROM, is $40. The current cost of 32K of CMOS RAM is about another $40.

The User Group would like to know of anyone who might be interested in buying one of these boards when they come into production. This will help in determining the quantities which we will order.

The prototype is up and running, however, if there is something else which you would specifically like to see on the board, (clock, RS423, etc) please let us know that too. May not be too late to change the layout.



```
750 DEF FNfield(string$,length%)
751 REM Takes string$, and outputs it
752 REM right justified, in a field
753 REM length% chractors long.
760 = RIGHT$( STRING$(length%," ")+st
    ring$ , length%)
```

The GRAPHICS ROM
from Computer Concepts.

Reviewed by Tim Ryan.   Cost approx $85.

I received my copy of the Graphics ROM as payment for programming services rendered, actually using the ROM. It was very welcome! I have now put it to a small amount of use, and thought that I should review it for others' information.

The first use I put it to was sprites, and I must say that other than the amount of memory that it gobbles (which is unavoidable) I was most impressed with the results. It is possible to have sprites in MODES 0, 1 and 2 only, which is a shame but does mean that the user has the most colours available.
My first sprites were of a tractor, and for those I used the largest available sprite in MODE 2 which is 24*24 pixels. The sprites can be sized as one, two, or three 'characters' in each direction (one character=8*8 pixels). I found it very easy to move the sprites, and with a little care was able to minimise the redrawing flicker as the sprites moved. It is of course impossible to totally eradicate flicker in a BASIC driven software sprite system but the flicker really was minimal.
There is one very interesting part to the sprites, and that is the ability to design several similar ones as an animation sequence, or 'film'. Once they are defined, by repeatedly putting up the sprite film number, the screen display cycles through the designed sequence automatically. An excellent animation effect can be achieved.

The next ROM routine I tried was the colour fill routine. Unfortunately, although this is a flood fill (i.e. it will fill any complex shape), it is rather slow. I know it is hard to write fast machine code fills for the Beeb as I have tried myself, but I would have thought it was possible to be faster than this routine. Apart from the speed, though, it certainly works perfectly.

There are many other parts to the ROM, some of which I have not yet tried out, others I have just touched on. I will list some of them with my comments:

3D drawing. This routine allows the user to specify a third coordinate in any plotting action which represents distance into the screen. This makes it possible to draw 3D objects and move them around the screen. Not only is DRAWing handled, but triangle fills also work. Looks most handy for 3D affects in games!

Big text. This routine allows big text to be drawn on the screen at any angle and direction (provided the ROTATE command has been used first to orientate the screen). It not only allows coloured text, but by using GCOLx,y where x>4 some interesting multicoloured patterns can be achieved. My only minor niggles are that the usual character set is used enlarged, which means they are rather chunky and the routine is rather slow (which suggests they are using standard OS calls!).

Turtle Graphics. I don't know much about this side of things, but I tried out all the commands, an the system seemed to work fast and neatly.

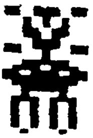
It is possible to redefine the screen coordinates in various ways such as rotate them, enlarge, or reduce them. This allows pictures to be replotted in different sizes and orientations.

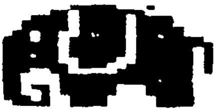
Computer Concepts has also provided a MODE 8, which gives 10 characters across the screen and only 80 pixels, but means that 16 colours are available with only 10K of memory!

There are one or two other drawing commands such as CIRCLE, and ARC, and PATTERN (polygon drawing), which are fairly self explanatory.

Well, as you can see, there is a lot to this ROM and even in the review I haven't covered everything, let alone used all the commands!

I feel the ROM is a good buy. Unfortunately I haven't seen the official ACORN one, so can't compare, but I know of two other people at least with this ROM in NZ, so there should soon be some software using it, floating around.





BEGINNERS SEMINAR

Anne F.

For every program idea, there are hundreds of different ways of expressing it. Let's just stick to BBC BASIC for now. Even so, there are still many ways to "sKin a cat".

Sandy W has sent in his program for an Electron which asKs for a colour number, then prints out the logical colour number and its name, in that colour. As he says, it was fun doing it, and it does worK. Here is the listing. To maKe it easy for you to see where it is going, lines have been drawn to show where the GOTOs go to and from:

```
   10 REM Electron Basic Colours
   20 REM by S Wigram
   30 MODE2
   40 *FX9,75
   50 *FX10,75
   60 INPUT'SPC1"WHAT COLOUR NUMBER?"'N
   70 IF N>=0 AND N<=15 THEN GOTO120
   80 IF N>=128 AND N<=143 THEN GOTO190
   90 COLOUR7:COLOUR128
  100 PRINT'"IS A WRONG NUMBER"''"PLEASE TRY AGAIN"
  110 GOTO 60
  120 COLOURN
  130 PRINT"TEXT COLOUR ";N:PRINT"(IN MODE 2)"
  140 FOR V=0 TO 15
  150   READ A$
  160   IF N=V PRINTSPC4"IS ";A$
  170   NEXT V
  180 GOTO250
  190 COLOURN
  200 M=N-128
  210 FOR W=0 TO 15
  220   READ A$
  230   IF M=W PRINT"BACKGROUND COLOUR";SPC2;N;" IS ";A$
  240   NEXT W
  250 RESTORE
  260 TIME=0
  270 REPEAT UNTIL TIME=175
  280 GOTO60
  290
  300 DATA BLACK,RED,GREEN,YELLOW,BLUE,MAGENTA,CYAN,WHITE
,BLACK/WHITE,RED/CYAN,GREEN/MAGENTA,YELLOW/BLUE,BLUE/YELL
OW,MAGENTA/GREEN,CYAN/RED,WHITE/BLACK
  310 DATA END
```

By way of contrast, here is another way of producing exactly the same thing on screen:

```
   10 REM  Colours
   20 REM  by me
   30 MODE2
   40 *FX9,75
   50 *FX10,75
   60 REPEAT
   70   REPEAT
   80     INPUT'SPC1"What colour number?"' N%
   90      IF (N%>=0 AND N%<=15) OR (N%>=128 AND N%<=143)
THEN ok% = TRUE ELSE ok% = FALSE:PROCwrong
  100     UNTIL ok%
  110   IF N%>15 THEN col% = N%-128 ELSE col% = N%
  120   COLOUR N%
  130   RESTORE
  140   FOR I% = 0 TO col%
  150     READ colour$
  160     NEXT I%
  170   IF N%<16 THEN screen$="text" ELSE screen$ = "back
ground"
  180   PRINTscreen$;" colour is"'colour$
  190   TIME=0:REPEAT UNTILTIME>175
  200   UNTIL FALSE
  210 END
  220
  230 DEF PROCwrong
  240 COLOUR128:COLOUR7
  250 PRINT';N%;" is a wrong number"''"Please try again"
  260 ENDPROC
  270 DATA black,red,green,yellow,blue,magenta,cyan,white
,black/white,red/cyan,green/magenta,yellow/blue,blue/yell
ow,magenta/green,cyan/red,white/black
```

There are several similarities and several differences:

1. All the GOTOs have been eliminated by using REPEAT...UNTILs and IF THEN ELSE. A lot of people forget that BBC BASIC has the ELSE available. More on this some other time.

2. VARIABLES

Listing 1:

N the colour number
V a counting loop to read data
M to use an altered version of "N"
W another counting loop to, read data

Listing 2:

N% the colour number that was input - it has to be a whole number, and integers (ie variables with a % after them) are processed faster by the computer. Whilst speed doesn't matter in this particular program, it's a good habit to get into using them.

I% a counting loop to read data. Every man and his dog uses I% for this purpose, so why buck the system?

col% an example of a "meaningful variable name" - it is perhaps more obvious that it is referring to a colour than "M"

The second counting loop for reading data was not needed because the two loops in I were combined in II with the introduction of the variable screen$

3. RENAMING VARIABLES

If you have more than one use for the input number, then it is a good idea to use different names if the value of that number is changed. This means that you will still maintain the original number no matter how you maltreat it, and also avoids confusion over what it means.

This is demonstrated in I line 200 and II line 110. If the input number was greater than 127 it will be a background colour number; but there isn't enough data in the DATA statements to read 128+ times through for the colour's name. Since a background colour number is the same as the foreground colour number plus 128, it makes sense to knock off that 128 before reading the data. Calling it M or col% also ensures that N or N% is still intact in case there are any other jobs for it to do (II line 170)

4. REMs

Believe it or not, in a year or two when you go back to a program, you will wish you had some of these to jog your memory quickly about what the program did, where it came from etc.

5. UPPER CASE VS LOWER CASE

BASIC commands must be in Upper case, so using lower case for other parts of the program contrasts more easily. Compare:

```
60 INPUT'SPC1"WHAT COLOUR NUMBER?"'N
80 INPUT'SPC1"What colour number?"' N%
```

6. Sandy stuck with the more legible idea of one statement per line. His original listing also had one line for each item of data. This is good as it makes it much easier to read through the program, make any alterations etc. To save magazine space however, the data were combined all into one line. Look also at line 90 in II. It was getting rather long. and was shortened by throwing a section out into PROCwrong. It could as easily have

read:

```
90 IF (N%>=0 AND N%<=15) OR (N%>=128 AND N%<=143) THEN ok% = TRU
E ELSE ok% = FALSE:COLOUR7:COLOUR128:PRINT';N%;" is a wrong numb
er"''"Please try again"
```

The idea behind setting up a PROC to handle wrong responses has these ideas behind it:

a) It keeps the line easier to read

b) a cursory reading of the listing doesn't want to get bogged down in the exact detail of how a wrong answer is handled

c) handling wrong responses might be required in some other part of an extended version of this program. By having it in a PROC it needs only to be typed in once in full.

7. <u>TIME</u>

The first listing wants a wait of 175/100 seconds before giving the next prompt. The second has changed it:

from UNTIL TIME = 175 to UNTIL TIME > 175

This is because you cannot guarantee that the computer will look at its clock at exactly 175 ticks. It may in fact only get around to looking at 174 and 176, but have been doing other things when 175 passed. Using the first method, if it didn't see 175 on its clock, it hasn't the brains to realise that if it sees 176, then 175 must have been reached, so it should go on to the next task. Instead it waits until the clock gets round to 175 again (takes about 35 weeks!) The second method ensures that if it misses 175, it will in fact go on when it sees more ticks than 175 have elapsed. In this program there should be no problem, but again, it is a good habit to get into, so you aren't caught on the hop when things like this matter.

8. <u>DATA END</u>

Here you don't need it because this program doesn't look for "END", and all the read loops look no further than 16 items, so will never get to the 17th.

BETA-BASE FILES

Steve Williams

Beta-base is a database product from Clares in the UK. Although in its standard form it is flexible enough for most applications, there are times where it simply won't do what is required. It is possible to write your own programs to access files created and modified by Beta-base if you know how the file is structured (read on!).

The file is in two parts and consists of a header portion followed by a data portion consisting of multiple fixed-length records. The header portion consists of 7 integers, followed by N sets of 1 byte/1 byte/Integer/String fields (where N is the number of fields in the record). Of the 7 integers, the first and the seventh are something to do with deleted records, while the others (in no special order) describe the following: Max number of records, number of records used, number of fields per record, Offset of first record from start of file (i.e. length of header data) and record length. For each field, data is as follows:

1 byte - field length (1-255)
1 byte - Field type (0=string,1=integer,2=real)
Integer - Offset of start of field within record
String - Field name

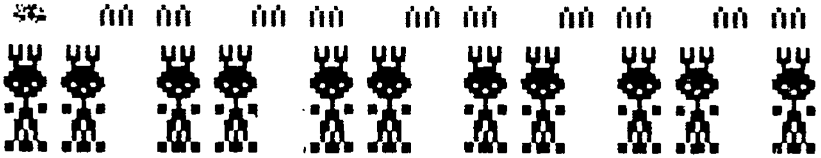
Deleted records have the string "*" written into each field position within the record.

The following program will read a specified field within a specified record for a specified filename (created by Beta-base). Note that in its present form it will not handle deleted records too well.

This could form the basis for a program which processed a Beta-base file sequentially in a way impossible using any of the standard options.

```
10REM REPORT PROGRAM
20REM STEVE WILLIAMS
30REM 31/10/85
40REM TO READ BETA-BASE FILES
50REM
55CLOSE#0
60MODE 7
```

```
   65INPUT"Filename " filename$
   70PROCsetupfile
   80CLS
   85REPEAT
   90INPUT "Enter record number " X%
  100INPUT "Enter field number " Y%
  105PRINT fieldprompt$(Y%);"/";
  110PROCreadfile(X%,Y%)
  120IF type$="INTEGER" THEN PRINT "INTEGER ";V%
  130IF type$="REAL" THEN PRINT "REAL ";V
  140IF type$="STRING" THEN PRINT "STRING ";V$
  150PRINT "Press any key...."
  160A=GET
  170UNTIL FALSE
 1000REM**********
 1010REM PROCsetupfile
 1020REM**********
 1030DEFPROCsetupfile
 1040M%=OPENIN(filename$)
 1050IF M%=0 THEN PRINT"File not found":STOP
 1060INPUT#M%,X%,lastrec%,fields%,maxrecords%,reclength%,offset%,X%
 1070DIM fieldlength%(fields%)
 1080DIM fieldtype%(fields%)
 1090DIM fieldoffset%(fields%)
 1100DIM fieldprompt$(fields%)
 1110FOR X%= 1 TO fields%
 1120fieldlength%(X%)=BGET#M%
 1130fieldtype%(X%)=BGET#M%
 1140INPUT#M%,fieldoffset%(X%)
 1150INPUT#M%,fieldprompt$(X%)
 1160NEXT X%
 1170ENDPROC
 1230REM**********
 1240REM PROCreadfile
 1250REM**********
 1260DEFPROCreadfile(recnum%,fieldnum%)
 1270PTR#M%=offset%+((recnum%-1)*reclength%)+fieldoffset%(fieldnum%)
 1280IF fieldtype%(fieldnum%)=0 THEN INPUT#M%,V$:type$="STRING"
 1290IF fieldtype%(fieldnum%)=1 THEN INPUT#M%,V%:type$="INTEGER"
 1300IF fieldtype%(fieldnum%)=2 THEN INPUT#M%,V:type$="REAL"
 1310ENDPROC
```



DON'T CLS

Tony Watton

Some screens deserve more than a hurried flash into oblivion, a little effort can make them linger as they depart. There are several methods to do this.Here is one.

```
  10  REM Alternative to CLS
  20  MODE 1
  30  FOR print=1 TO 30
  40    PRINT"Don't just CLS it...wave a window at it"
  50    NEXT
  60  FOR taihoa = 1 TO 1000:NEXT
  70  FOR A = 31 TO 1 STEP-1
  80    VDU 28,0,A,39,A-1
  90    CLS
 100    NEXT
 110  VDU 26
 120  END
```

Line 80 sets up a text window across the bottom of the screen and clears it while moving it upwards (70 to 100). Changing lines 70 and 80 (below) changes the window to one that expands from the top left corner.

```
  70  FOR A = 0 TO 31
  80    VDU 28,0,A,A+8,0
```

It is possible to have more than one window and colour as well. Change and/or insert lines 70 to 95. Obviously there are many possible combinations and variations and there should be one to suit your needs.

```
  70  FOR A = 0 TO 38
  80    VDU 28,A,15,A+1,0
  85    COLOUR 129:CLS
  90    VDU 28,38-A,31,39-A,16
  95    COLOUR 130,CLS
```

* * *

CHRISTMAS...

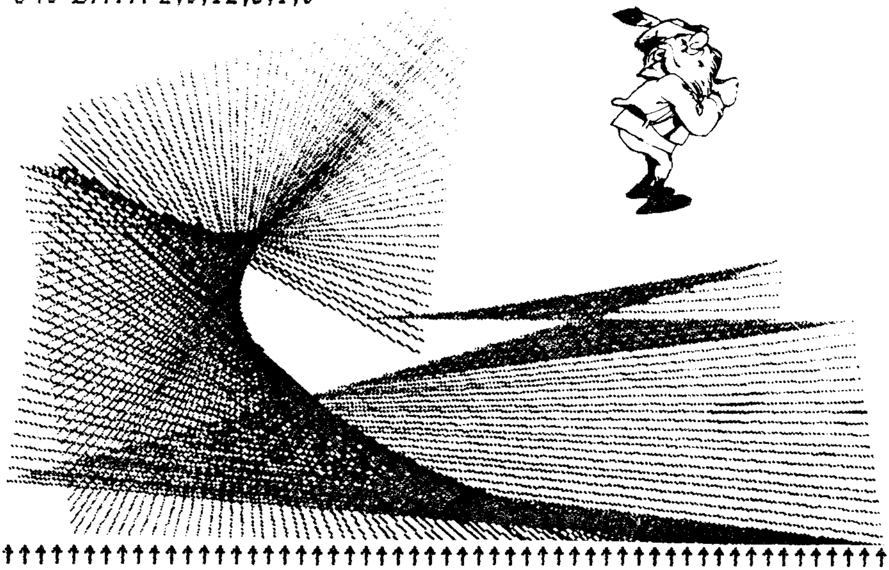
WITH STRINGS.

Anne F.

You are all familiar with "Jingle Bells" resounding through the shops at this time of year. Now you can bring it in to your homes with this program. Not only does it play the tune, it also draws the reins for Santa's reindeer.

```
   10 REM    JINGLE BELLS
   20 REM   with String Art
   30 REM
   40 REM Nov '82 rev. Nov'85
   50 REM
   60 ENVELOPE1,3,0,0,0,0,0,0,121,-10,-5,-2,120,120
   70 REPEAT
   80   *fx15
   90   tune_end% = FALSE
  100   RESTORE
  110   MODE0
  120   PROCstartstring
  130   PROCstring
  140   REPEAT UNTIL ADVAL (-7) > 14
  150   UNTIL FALSE
  160 END
  170
  180 DEFPROCstartstring
  190 X%=RND(1280):Y%=RND(1280)

  200 L%=RND(1280):M%=RND(1024)
  210 U%=20-RND(40):V%=20-RND(40)
  220 P%=20-RND(40):Q%=20-RND(40)
  230 ENDPROC
  240
  250 DEFPROCstring
  260 REPEAT
  270   time% = TIME + 5
  280   MOVE X%,Y%
  290   DRAW L%,M%
  300   IF X%+U%>1279 OR X%+U%<0 U%=-U%
  310   IF Y%+V%>1023 OR Y%+V%<0 V%=-V%
  320   IF L%+P%>1279 OR L%+P%<0 P%=-P%
  330   IF M%+Q%>1023 OR M%+Q%<0 Q%=-Q%
  340   X%=X%+U%:Y%=Y%+V%
  350   L%=L%+P%:M%=M%+Q%
  360   REPEAT
  370     IF ADVAL(-6)>0 AND ADVAL(-7)>0 THEN PROCtune
  380    UNTIL TIME>time% OR tune_end%
  390   UNTIL tune_end%
  400 ENDPROC
  410
  420 DEF PROCtune
  430 LOCAL chan%,pitch%,dur%
  440 READ chan%,pitch%,dur%
  450 IF chan%=3 THEN tune_end%=TRUE ELSE SOUND chan%,1,
pitch%,dur%
  460 ENDPROC
  470
  480  DATA  1,137,6,2,73,6,1,137,6,2,89,6,1,137,12,2,101,6,2,89,6
,1,137,6,2,73,6,1,137,6,2,89,6,1,137,12,2,101,6,2,89,6,1,137,6,2
,73,6,1,149,6,2,89,6,1,121,9,2,101,6,1,129,3,2,89,6,1,137,24,2,7
3,6,2,89,6,2,101,6,2,73,6
  490  DATA  1,141,6,2,93,6,1,141,6,2,109,6,1,141,9,2,73,6,1,141,3
,2,73,6,1,141,6,2,109,6,1,137,6,2,89,6,1,137,6,2,101,6,1,137,3,2
,73,6,1,137,3
  500  DATA  1,137,6,2,109,6,1,129,6,2,93,6,1,129,6,2,109,6,1,137,
6,2,73,6,1,129,12,2,101,6,2,117,6,1,149,12,2,129,6,2,117,6
  510  DATA  1,137,6,2,73,6,1,137,6,2,89,6,1,137,12,2,101,6,2,89,6
,1,137,6,2,73,6,1,137,6,2,89,6,1,137,12,2,101,6,2,89,6,1,137,6,2
,73,6,1,149,6,2,89,6,1,121,9,2,101,6,1,129,3,2,89,6,1,137,24,2,7
3,6,2,89,6,2,101,6,2,73,6
  520  DATA  1,141,6,2,93,6,1,141,6,2,109,6,1,141,9,2,73,6,1,141,3
,2,73,6,1,141,6,2,109,6,1,137,6,2,89,6,1,137,6,2,101,6,1,137,3,2
,73,6,1,137,3
  530  DATA  1,149,6,2,69,6,1,149,6,2,81,6,1,141,6,2,101,6,1,129,6
,2,69,6,1,121,6,2,73,6,2,101,6,2,73,18
  540 DATA 2,0,12,3,1,0
```



3D TEXT

Neville T.

Here at last is Neville Tomlinson's program which produces rather stunning 3D visual displays ideal for headings, titles, book covers etc. Neville says that he has had a lot of fun showing it off to friends and providing them with printouts of their names:

The program begins by asking you what sort of background you would like drawn - stars or atoms. You then input the data for each star/atom to be drawn. (X,Y coordinates of the centre, size in the form of inner and outer circle radii). The program is capable of handling up to three lines of text. Each line of text input may have an associated 'viewing distance' - the default is 10 but you an enter anything from 4 up. A viewing distance of 4 gives about one letter to a screen! The larger the viewing distance, the more text that can be fitted in - I generaly use a viewing distance of about 12. If you find that the text is not centered nicely then just precede the line with a few blank spaces when you enter it.

The program sets up function keys to produce characters 224 - 232. So, if you want to produce a 3D graphics characters (or include one in your line) just define it in the normal way (VDU23) then enter it by pressing the appropriate function key.

The next stage asks you whether you would like a logo. This is just a little four line message displayed in the right hand lower corner of the screen. The logo text is automatically centered - and again graphics characters may be used.

Once the picture has been drawn, you can start again by pressing return. If you want a screendump, add a screendump in lines 70-90, then call it at 440. Now evey time you press "P" the screen will be dumped. Press Q to quit, the <RETURN> key to rerun.

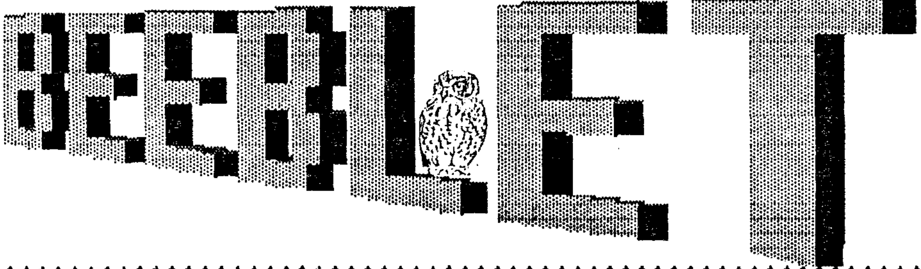
To get you started, try entering two background atoms - both centred at 640,512. Enter one with outer radius 500, inner radius 200 : the other with an outer radius of 500, inner radius 50. Try a single line of text over this, like "__Hello" (_ denotes a space), with a viewing distance of 11. Anyway, you are bound to have a lot of fun with this program.

For those of you who want to know a little more about the program, a few brief comments:

The PROCs PROCstar and PROCatom are used to draw a single star/atom. Details for each are held in the bgd() - these details are entered in PROCatomstar. In these PROCS X% and Y% are the centre coordinates, R1 is te outide radius, R2 the inside radius, and (in PROCstar) N% is the number of points. You probably recognise PROCstar from an earlier edition of Beelet. Both these PROCs are completely self contained and could easily be cribbed for other programs.

Lines 260-280 form a REPEAT .. UNTIL loop which draws the background. The finished background is signified by the value 9.9 as the next element of the bgd array. There is a maximum of nine stars/atoms which can be drawn. If you want more, increase the DIM in line 120 and the variable pnt% in line 740, which just counts how many STAR/ATOMS have been entered.

The text is entered in PROCtext, the logo in PROClogo. PROCsetup does the auto-centering and printing of the logo. Lines 380-400 call the procedure to draw the 3D text for each line. The distance down the screen is determined by the last(?) parameters. The top of the screen is 0, the bottom somewhere around -2000. The real "meat" of the program is PROCdraw called with three parameters, W$ (text to be printed), vd% (viewing distance entered in PROCtext) and HEIGHT (the position on the screen, 0 to -2000). This Proc steps through each character in the line of text. If the character is undefined, the definition is read from page &C00, otherwise the definition is read from the ROM (starting at &BF00). Offsets are calculated from the ASCII valuea of the characters. The definition is stepped through from bottom to top, each "dot" of the character being printe in 3D block. Each face is a different colour. The overall effect is achieved by printing "blocks" on top of or beside each other (as you will see by watching the program draw text.) PROCcube draws a single "block", the three faces being drawn by PROCfront, PROCtop and PROCside. The complicated looking line 1170 is automatic scaling. In the first parameter of PROCcube, the -3000 is the offset from the RHS of the screen. Well, that's about it - go design some 3D headings of your own!



```
10 REM 3D Titles
20 REM by Neville Tomlinson
30 REM May 1985
40 :
50 ON ERROR MODE7:IF ERR=17 RUN ELS
   E REPORT:PRINT" at line ";ERL:EN
   D
60 :
70 REM If you will be dumping to a
80 REM printer, *LOAD the screen du
   mp
90 REM routine here.
100 :
110 *TV255
120 DIM bgd(10,4)
130 *FX225,224
140 REPEAT
150   bgd(0,0)=9.9
160   MODE7:FOR X%=2TO3:PRINT TAB(6,
       X%)CHR$134CHR$157CHR$132CHR$
       141SPC8"3D Text"SPC10CHR$156
       :NEXT
170   VDU28,0,24,39,4
180   PRINT'''CHR$129"Backgrounds:"'
     'TAB(5)"1. None"'TAB(5)"2. Sta
     r"'TAB(5)"3. Atom"
190   PRINT'''CHR$132"Which Option :
      ";:REPEAT:backgrd=GET-48:UNTIL
      backgrd>0 AND backgrd<4
200   CLS:IF backgrd>1 PROCatomstar
210   MODE1:COLOUR3:PRINT TAB(16,3)"
     3D Text":VDU28,0,31,39,4
220   PROCtext:CLS
230   PRINT''''"Do you want a logo ?
      ";:REPEAT:logo$=GET$:UNTIL IN
     STR("YN",logo$)
240   IF logo$="Y" PROClogo
250   :
260   MODE1:VDU19,1,3;0;19,2,2;0;19,
     3,4;0;
270   IF backgrd>1 PROCbackground
280   VDU29,0;0;
290   IF logo$="Y" PROCsetup
300   :
310   VDU29,440;900;:ANGLE=60:S%=200
     0
320   Sin=SINRAD(-ANGLE):Cos=COSRAD(
     -ANGLE)
330   :
340   REM Third Parameter of PROCdra
     w is
350   REM distance down the screen.
360   REM 0=top, -2000=bottom
370   :
380   IF LENA$>0 PROCdraw(A$,D1*-500
     ,0)
390   IF LENB$>0 PROCdraw(B$,D2*-500
     ,-800)
400   IF LENC$>0 PROCdraw(C$,D3*-500
     ,-1600)
410   :
420   REPEAT
430     A=GET:IF A=13 RUN
440     IF A=ASC"P": REM Call Screen
        Dump
450     UNTIL A=ASC"Q"
460   UNTIL A=ASC"Q"
470 END
480 :
490 DEFPROCsetup
500 K=LEND$:IF LENE$>K K=LENE$
510 IF LENF$>K K=LENF$
520 IF LENG$>K K=LENG$
530 D$=STRING$((K-LEND$)/2," ")+D$:E
    $=STRING$((K-LENE$)/2," ")+E$:F$
    =STRING$((K-LENF$)/2," ")+F$:G$=
    STRING$((K-LENG$)/2," ")+G$
540 X%=1280-K*32
550 VDU5:MOVE X%,140:PRINT D$:MOVE X
    %,105:PRINT E$:MOVE X%,70:PRINT
    F$:MOVE X%,35:PRINT G$:VDU4
560 ENDPROC
570 :
580 DEFPROCbackground
590 pnt%=0:GCOL0,2:REPEAT
600   IFbackgrd=3 PROCatom(bgd(pnt%,
     0),bgd(pnt%,1),bgd(pnt%,2),bgd
     (pnt%,3)) ELSE PROCstar(bgd(pn
     t%,0),bgd(pnt%,1),bgd(pnt%,2),
     bgd(pnt%,3),bgd(pnt%,4))
610   pnt%=pnt%+1:UNTIL bgd(pnt%,0)=
    9.9
620 ENDPROC
630 DEFPROCatomstar:pnt%=0
```

```
640 PRINT''TAB(14)CHR$129"Draw A";:I
    F backgrd=3 PRINT"n Atom" ELSE P
    RINT" Star"
650 VDU28,0,24,39,8:REPEAT:CLS:PRINT
      TAB(10,5)CHR$133"Centre X"''T
     AB(10)CHR$133"Centre Y"''TAB(1
     0)CHR$133"Outer radius"''TAB(1
     0)CHR$133"Inner radius"
660   IF backgrd=2 PRINT'TAB(10)CHR$
     133"No. of points"
670   INPUT TAB(25,5)X%:IF X%<0 OR X
     %>1280 X%=640
680   INPUT TAB(25,7)Y%:IF Y%<0 OR Y
     %>1024 Y%=512
690   INPUT TAB(25,9)R1%:IF R1%>600
     R1%=500
700   INPUT TAB(25,11)bgd(pnt%,3)
710   IF backgrd=2 INPUT TAB(25,13)b
     gd(pnt%,4)
720   bgd(pnt%,0)=X%:bgd(pnt%,1)=Y%:
     bgd(pnt%,2)=R1%
730   pnt%=pnt%+1:INPUT TAB(10,16)"D
     raw another Y/N ?":REPEAT:K$=G
       ET$:UNTIL INSTR("YyNn",K$)
740   UNTIL INSTR("Nn",K$) OR pnt%=9
750 bgd(pnt%,0)=9.9:VDU28,0,24,39,4
760 ENDPROC
770 :
780 DEFPROCtext
790 CLS:COLOUR3:PRINT TAB(15,2)"TEXT
     ENTRY:"
800 COLOUR1:PRINT'''"10 is default V
    iewing Distance"''"The larger th
    e Viewing Distance, the"'"more t
    ext you can fit in"':COLOUR2:PRI
    NT"Function keys have been set t
    o graphics characters..."
810 COLOUR1:PRINT''"Line 1 : ";:INPU
    TLINE A$:COLOUR2:PRINT"Viewing D
    istance ";:INPUT D1:IF D1<4 D1=1
    0
820 COLOUR1:PRINT''"Line 2 : ";:INPU
    TLINE B$:COLOUR2:PRINT"Viewing D
    istance ";:INPUT D2:IF D2<4 D2=1
    0
830 COLOUR1:PRINT''"Line 3 : ";:INPU
    TLINE C$:COLOUR2:PRINT"Viewing D
    istance ";:INPUT D3:IF D3<4 D3=1
    0
 840 ENDPROC
 850 :
 860 DEFPROClogo
 870 COLOUR0:PRINT TAB(15,2)"LOGO ENT
     RY:"''':COLOUR1:PRINT"Function k
     eys have been set to graphics ch
     aracters..."':COLOUR2:PRINT"Auto
     matically Centred"
 880 COLOUR3:PRINT''"Line 1 : ";:INPU
     TD$:PRINT'"Line 2 : ";:INPUTE$:P
     RINT'"Line 3 : ";:INPUTF$:PRINT'
     "Line 4 : ";:INPUTG$
 890 ENDPROC
 900 :
 910 DEFPROCstar(X%,Y%,R1,R2,N%)
 920 LOCAL A,D,I%:VDU29,X%;Y%;
 930 MOVE R1,0:A=0:D=PI/N%
 940 FOR I%=0TON%:A=A+D:DRAW R2*COSA,
      R2*SINA
 950   A=A+D:DRAW R1*COSA,R1*SINA
 960   NEXTI%
 970 ENDPROC
 980 :
 990 DEFPROCatom(X%,Y%,R1,R2)
1000 LOCAL PS,PC,NS,NC
1010 PS=SINRAD45:PC=COSRAD45:NS=SINRA
     D-45:NC=COSRAD-45
1020 MOVE X%+R2,Y%
1030 FOR A=0TO360 STEP15:DRAW X%+R2*C
     OSRADA,Y%+R1*SINRADA:NEXT
1040 MOVE X%+R1,Y%
1050 FOR A=0TO360 STEP15:DRAW X%+R1*C
     OSRADA,Y%+R2*SINRADA:NEXT
1060 MOVE X%+R2*PC,Y%+R1*PS
1070 FOR A=0TO360 STEP15:DRAW X%+R2*C
     OSRADA*PC-R1*SINRADA*PS,Y%+R1*
     COSRADA*PS-R2*SINRADA*PC:NEXT
1080 MOVE X%+R2*NC,Y%+R1*NS
1090 FOR A=0TO360 STEP15:DRAW X%+R2*C
     OSRADA*NC-R1*SINRADA*NS,Y%+R1*
     COSRADA*NS-R2*SINRADA*NC:NEXT
1100 ENDPROC
1110 :
1120 DEFPROCdraw(W$,VD%,HEIGHT)
1130 FOR H%=1TOLENW$
```

```
1140   IF ASCMID$(W$,H%,1))223 L%=&C0
    0+8*(ASCMID$(W$,H%,1)-224):GOT
    01160
1150   L%=&BF00+8*ASCMID$(W$,H%,1)
1160   FOR I%=7TO0 STEP-1
1170     FOR J%=7TO0 STEP-1
1180       IF (L%?I%AND2^J%) PROCCUBE
        (-3000-((VD%(-5000)*(VD%+5
        000)*.5)+800*H%-J%*100,HEI
        GHT-I%*100,0)
1190         NEXT
1200       NEXT
1210   NEXT
1220 ENDPROC
1230 :
1240 DEFPROCCUBE(X%,Y%,Z%)
1250 PROCFRONT(X%,Y%,Z%)
1260 PROCSIDE(X%+100,Y%,Z%)
1270 PROCTOP(X%,Y%+100,Z%)
1280 ENDPROC
1290 :
1300 DEFPROCFRONT(X%,Y%,Z%)
1310 RESTORE1330:PROCFACE(1)
1320 ENDPROC
1330 DATA4,0,0,0,4,100,0,0,85,0,100,0
     ,85,100,100,0
1340 :
1350 DEFPROCSIDE(X%,Y%,Z%)
1360 RESTORE1380:PROCFACE(3)
1370 ENDPROC
1380 DATA4,0,0,0,4,0,0,100,85,0,100,0
     ,85,0,100,100
1390 :
1400 DEFPROCTOP(X%,Y%,Z%)
1410 RESTORE1430:PROCFACE(2)
1420 ENDPROC
1430 DATA4,0,0,0,4,0,0,100,85,100,0,0
     ,85,100,0,100
1440 :
1450 DEFPROCFACE(N)
1460 LOCALI%
1470 GCOL0,N
1480 FOR I%=1TO4:READ K%,A%,B%,C%
1490   PROCP(K%,X%+A%,Y%+B%,Z%+C%)
1500   NEXT
1510 ENDPROC
1520 :
1530 DEFPROCP(K%,X%,Y%,Z%)
1540 x%=X%*Cos-Z%*Sin:z%=Z%*Cos+X%*Si
   n
1550 y%=Y%
1560 M=S%/(z%-VD%):PLOT K%,x%*M,y%*M
1570 ENDPROC
```

* * *

BEEBLETS IN A HEAP?

BIND THEM TO YOU...

with a Binder from the User Group.

Only $2.50 +65c p&p

Need holes punched in magazine,
them will hold many a year's issues.

REHASH

Neil Gordon

Last month a previously unknown poem by William Shakespeare was discovered gathering dust in a library in England. How did they know it was by Shakespeare? They would have compared the style, counts of specific words and word sequences used, and so on, against those of other works by Shakespeare.

Now you, too, can generate Shakespearean works. All you need to do is provide some text (say, a sonnet or two) for this REHASH program and it will regurgitate new text that has strong style similarities to the original, even if it doesn't make a lot of sense.

The technique is based on something I read in "Byte" magazine earlier this year. The idea is only to output strings of characters that occur in the original text. If the matching strings are short (say just 2 characters) then the results will be rather un-English like. However, if the strings are 4 or 5 characters in length, the output text contains mostly proper English words with a few new conglomerates. A length of 7 characters results in phrases appearing, and with 10 characters large sections of the original text are reproduced.

We first load the original text file into memory, then choose a random place to start. The next 'nmatch' characters are saved as a string down in zero page (starting at 'word'). For each new character produced, we start at a new random place in the text and look for a match to the current string. When we find it, we output the next character in the text, add it to end of our string and drop the first character of the string.

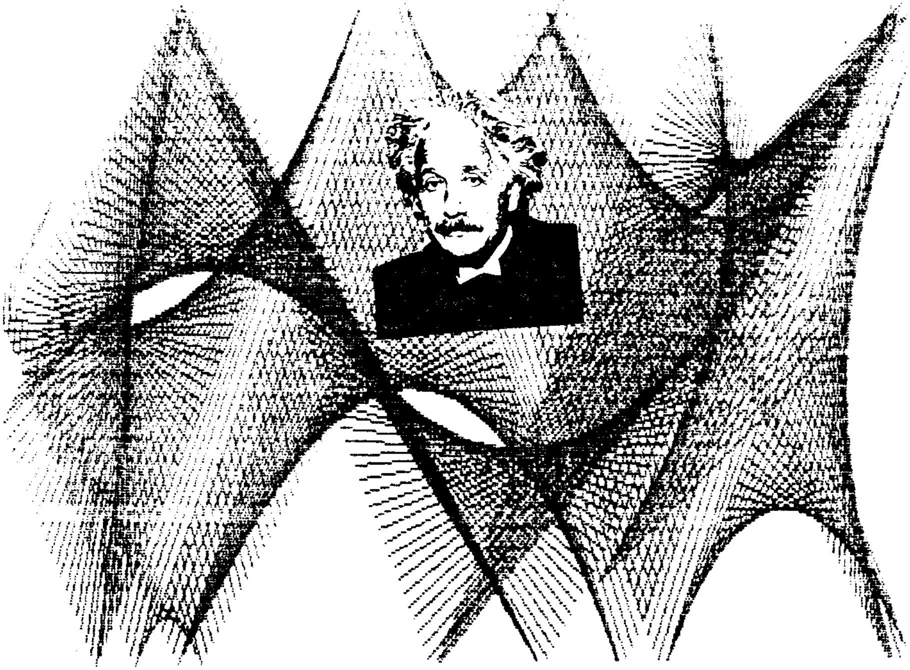
I originally tried doing the search for a match in BASIC, but it was so painfully slow that I had to resort to the machine code version given in the program. The main awkward bit is wrapping around to the start of the text if you don't find a match before reaching it. Looping can be a problem for a couple of reasons that I won't go into, but I think I've trapped all the possibilities.

The text produced will be written to the (MODE 7) screen. You might like to switch to MODE 3 to make it easier to read if you can still fit the original text into memory. On the off chance that you think you might produce a masterpiece you can choose to write the output to a new file. When you get tired of the current output, press any key and you get another go at the same text (try using a different match).

As an example of the output, the following short segment was produced with a 5 character match from my President's Report in this issue, and is totally recognisable as such:

```
The results resoundingly favour of the questionnaires that we
were looking for replies to each side, the question, the
accumulated surplus to each option.  The bimonthly newsletter,
Bits and frequent questionnaires.  This bar replies to all those
ticked 'Don't Know' or, presumably not make it.  Adding up those
ticked 'Don't Know' or, presumably not Knowing members, rather
the popularity stakes, with the bottom of them replies to have
been misinteresting of tardy renewals.
```

As you can see, the results are rather entertaining - I hope you have fun with the program. Among other things, you can also use it on *SPOOLed BASIC programs to write new programs - no guarantee they'll work though! REHASH should be a boon to the editor too - in case of lack of material for "Beeblet" just take any old article and feed through REHASH for instant new article.



```
 10 REM Rehash an existing text file
 20 REM (c) Neil Gordon - December 19
    85
 30 MODE 7 : REM Could be MODE 3 if i
    nput text file is small enough
 40 :
 50 REM Get file name and load into m
    emory
 60 :
 70 REPEAT
 80   PRINT "Enter Input Text File Na
      me: ";
 90   INPUT file$
100   channel=OPENIN(file$)
110   UNTIL channel>0
120 :
130 size=EXT#channel
140 CLOSE#channel
150 DIM command 50,code 100,text size
160 $command="LOAD "+file$+" "+STR$~t
    ext
170 X%=command MOD 256
180 Y%=command DIV 256
190 CALL &FFF7 : REM OSCLI
200 :
210 REM Loop over rehash attempts
220 :
230 REPEAT
240   PRINT''
250   IF FNask("Write output to a new
       file") THEN REPEAT INPUT "Outp
        ut file name: "file$ : channe
        l=OPENOUT(file$) : UNTIL chan
      nel>0 ELSE channel=0
260   REPEAT
270     INPUT "Match how many charact
        ers? (1 to 10): " nmatch
280     UNTIL nmatch>0 AND nmatch<11
290   :
300   nmatch=nmatch-1
310   length=size-nmatch
320   PROCassemble
330   :
340   REM Start off in random positio
      n
350   :
360   PROCstartup
370   REPEAT
380     last=!pointer
390     !pointer=text+RND(length)
400     CALL code
410     IF ?char=255 THEN PROCstartup
         : ?char=13
420     IF ?char>127 THEN VDU 32 ELSE
         IF ?char>31 THEN VDU ?char E
        LSE IF ?char=13 THEN VDU 13,1
        0
430     IF channel<>0 THEN BPUT#chann
        el,?char
440     IF last=!pointer THEN ?(!poin
        ter)=0 : REM Avoid loops
450     UNTIL INKEY(0)>0
460   CLOSE#channel
470   UNTIL FALSE
480 END
490 :
500 DEF FNask(question$)
510 LOCAL char
520 PRINT question$+" (Y/N):";
530 REPEAT
540   char=GET
550   IF char=ASC"Y" OR char=ASC"y" T
      HEN PRINT "Yes" : char=1 ELSE I
      F char=ASC"N" OR char=ASC"n" TH
      EN PRINT "No" : char=0 : ELSE c
      har=2
560   UNTIL char<>2
570 =(char=1)
580 :
590 DEF PROCstartup
600 !pointer=text+RND(length)
610 FOR I%=0 TO nmatch
620   string?I%=?((!pointer)+I%)
630   NEXT
640 ENDPROC
650 :
660 DEF PROCassemble
670 pointer=&70
680 char=&74
690 count=&75
700 string=&76
710 :
720 FOR pass=0 TO 2 STEP 2
```

```
730   P%=code
740   [OPT pass
750           LDA #254
760           STA count
770 .trynext
780           LDY #0
790 .loop_Y
800           LDA (pointer),Y
810           CMP string,Y
820           BNE nomatch
830           INY
840           CPY #(nmatch+1)
850           BMI loop_Y
860           LDA (pointer),Y
870           STA char
880           TAX
890 \ Move string down
900           LDY #0
910 .loop
920           LDA string+1,Y
930           STA string,Y
940           INY
950           CPY #nmatch
960           BNE loop
970           STX string,Y
980           RTS
990

1000 .nomatch
1010          CLC
1020          LDA pointer
1030          ADC #1
1040          STA pointer
1050          LDA pointer+1
```

```
1060          ADC #0
1070          STA pointer+1
1080          CMP #((text+length)DIV
             256)
1090          BMI ok
1100          LDA pointer
1110          CMP #((text+length)MOD
             256)
1120          BMI ok
1130          LDA #(text MOD 256)
1140          STA pointer
1150          LDA #(text DIV 256)
1160          STA pointer+1
1170          INC count
1180          BNE ok
1190

1200 \ Oops - looping (probably stuck
             at end of text)
1210 \ Indicate by a strange character
1220 \ and let BASIC fix it up
1230

1240          LDA #255
1250          STA char
1260          RTS
1270

1280 .ok
1290          JMP trynext
1300

1310          ]
1320  NEXT pass
1330 ENDPROC
```

FINDING STRINGS AND BASIC KEYWORDS

WHERE and *FVAR explained

Don Austin

NAME of program: WHERE (source program)
*FVAR (Find VARiable, machine code)

PURPOSE: To list the line numbers and the content of the line(s) wherever the target string or token is found.

SETTING UP: Type in the listing below. If the listing is correct, line 1720 will tell you. When correct SAVE the listing as "WHERE" and keep it filed away. Then *SAVE the machine code as shown in line 1720.

DETAIL: All users

To use the program, just type *FVAR (upper or lower case). This will load the program into page &900 which will NOT disturb your BASIC program. Once loaded the program will initialise itself and let you know it's ready to use it with the prompt:

```
Syntax: LINE <string/target value>
Set *FX225,<Keyword token in decimal> Press key f0 for target
value
```

It is not necessary to type *FVAR each time. Once the prompt has appeared, just use the *LINE command whenever needed.

Cassette users could either change the &A00 in lines 100, 1620 to &C00 if they are not using the user definable characters or set PAGE to &1000 and change the &900 to &E00 and then *RUN FVAR.

Finding words that are recognised by BBC BASIC is made tricky by the fact that BASIC does not store keywords like PRINT as a word at all but as a single byte called a "token". Thus to find a BASIC token such as PRINT, it is necessary to know what its byte value is. That is the purpose of the procedure "PROCtokens" in line 40. It lists all the BASIC keywords and their values in decimal. Two have no keywords, that's right.

* * *

Two examples

To find a string

(1)First of all, get the program from disk - type *FVAR and press RETURN. You see:

```
>*FVAR

Syntax: LINE <string/target value>
Set *FX225,<Keyword token in decimal>
Press key f0 for target value
```

(2)Say the wanted string is the variable "flag%", so type *LINE flag% and press RETURN. You see:

```
>*LINE flag
```

* * *

GETTING THE MOST FROM YOUR BBC MICRO

- Clive Williamson
- 208 pages (price unknown)

This book is subtitled "The indispensable guide to your home computer", a fairly arrogant assertion. In it's 200-odd pages, the book attempts to cover every topic possible, from how to plug all the bits together to a description of how Bitstik works. Inevitably it fails to cover any single area in detail and I suspect that the author has tried to do too much. That's not to say that what is there is not well-presented. There is a section on further reading, and a list of contact addresses (user groups etc), both very useful.

This book duplicates a lot of what appears in the User Guide, (in arguably a more readable form), and in all honesty I could not recommend it to anyone bar the absolute novice.

* * *

```
   10 REM WHERE finds variable or token
       in a BASIC listing, giving the l
      ine
   20 REM number(s) and the line conten
      ts  (C) Don Austin, Tokoroa NZ, 1
      985
   30 MODE0
   40 PROCtokens
   50 MODE3
   60 osbyte=&FFF4:osrdch=&FFE0:oswrch=
      &FFEE:osnewl=&FFE7:osword=&FFF1
   70 base%=&70:len%=&72:flag%=&73:line
      %=&74:linelen%=&76:start%=&77
   80 found%=&79:asc%=&7A:end%=&7B:real
      len%=&7E:text%=&8F:temp%=&80
   90
  100 FORK%=0TO3STEP3:P%=&900
  110   [OPTK%
  120 .start
  130             LDA #line MOD256
  140             STA &200
                   \ Set USERV to point t
                  o start of routine
  150             LDA #line DIV256
  160             STA &201
  170             BRK
  180             EQUB 255
  190             EQUS"Syntax  LINE (stri
                  ng/target value)"
  200             EQUB 10
                  EQUB 10
                  EQUB 13
  210             EQUS"Set *FX225,(keywor
                  d token in decimal)"
  220             EQUB 10
                  EQUB 13
  230             EQUS"Press key f0 for t
                  arget value"
                  EQUB 13
  240             BRK
  250 .line
  260             CMP #1
  270             BEQ code
  280             RTS
  290 .code
  300             STX text%
                   \ X and Y point to th
                  e string
  310             STY text%+1
  320             LDA #14
                   \ Set page mode
  330             JSR oswrch
  340             LDA &18
                   \ Find PAGE
  350             STA base%+1
  360             LDY #0
  370             STY base%
  380 .text
  390             LDA (text%),Y
                   \ Get the string
  400             INY
                   \ and find the LEN
  410             CMP #127
  420             BCS token
                   \ If value > 127, it
                  must be a token and LEN
                  =1
  430             CMP #13
  440             BNE text
  450             DEY
  460 .token
  470             STY len%
                   \ Store the LEN
  480             BEQ done
                    \ no string so retur
                  n to BASIC
  490             JSR &850E
                   \ PRINT the expanded
                  token to confirm
  500             JSR osnewl
  510
  520 .loop2
  530             LDY #1
  540             STY flag%
                   \ Set flag to not rep
                  eat the line listing
  550             LDA (base%),Y
                   \ Get the current lin
                  e number
  560             STA line%+1
  570             INY
  580             LDA (base%),Y
  590             STA line%
```

```
600            INY
610            LDA (base%),Y
                 \ Get the line's leng
               th
620            STA reallen%
630            SEC
640            SBC #4
                 \ Subtract 4
650            STA linelen%
                 \ to remove the non-c
               haracter bytes
660            CLC
670            LDA base%
680            ADC #4
690            STA start%
                 \ Add 4 to get to the
                start of the line's
700            LDA base%+1
                 \ contents
710            ADC #0
720            STA start%+1
730
740 .search                      \ Start
               of the search routine
750            LDX #0
                 \ Set the X counter
760            STX temp%
                 \ and clear these tem
               porary locations
770            STX temp%+1
780 .again
790            LDY #&FF
800            STY found%
                 \ Set the found flag
               = TRUE
810            INY
                 \ Set the Y counter =
                0
820 .loop3
830            LDA (text%),Y
                 \ Get the Yth char of
                the string
840            STA asc%
850            STY temp%+1
                 \ Remember the Y valu
               e
860            LDA temp%
                 \ Get the X value
870            CLC
880            ADC temp%+1
                 \ Add the X value to
                the Y value
890            TAY
900            LDA (start%),Y
                 \ Get the (X+Y)th cha
               r of the string
910            CMP asc%
                 \ Are they the same ?
920            BEQ equal
930            LDA #0
940            STA found%
                 \ Not found, so found
               -flag = FALSE
950 .equal
960            LDY temp%+1
                 \ Restore the Y value
970            INY
980            CPY len%
                 \ and see if whole st
               ring has been checked
990            BNE loop3
1000            STX temp%+3
                 \ Remember the X valu
               e in this loop
1010            LDA found%
                 \ Is the string found
                ?
1020            BEQ notfound
1030            LDA flag%
                 \ Has the line alread
               y been PRINTed ?
1040            BEQ skip
1050            JSR printout
                 \ PRINT the line numb
               er and contents
1060 .notfound
1070            LDX temp%+3
                 \ Restore the X value
                for this loop
1080            INX
               STX temp%            \
               Update for the search l
               oop (X+Y)
1090            CPX linelen%
                 \ Finished this line
               ?
```

```
1100            BNE again
1110 .skip
1120            CLC
1130            LDA base%
                 \ Update for the next
                line
1140            ADC reallen%
1150            STA base%
1160            LDA base%+1
1170            ADC #0
1180            STA base%+1
1190            LDY #1
1200            LDA (base%),Y
1210            CMP #&FF
                 \ Check for the end o
               f program marker
1220            BNE loop2
                 \ If not found, repea
               t the loop
1230 .done
1240            LDA #15
                 \ Turn off page mode
1250            JSR oswrch
1260            LDA #225
                 \ Reset the user-keys
                back to default value
1270            LDX #1
1280            LDY #0
1290            JSR osbyte
1300            RTS
                 \ Return to BASIC
1310 .printout
1320            LDX #0
1330 .convert
1340            LDY #16
                 \ PRINT the 2 byte li
               ne number
1350            LDA #0
1360 .div10                        \ as a d
               ecimal number
1370            ASL line%
1380            ROL line%+1
1390            ROL A
1400            CMP #10
1410            BCC less
1420            SBC #10
1430            INC line%
1440 .less
1450            DEY
1460            BNE div10
1470            CLC
1480            ADC #&30
1490            PHA
1500            INX
1510            LDA line%
1520            ORA line%+1
1530            BNE convert
1540 .printnum
1550            PLA
1560            JSR oswrch
1570            DEX
1580            BNE printnum
1590            STX flag%
                \ Set the line listed
               flag = FALSE
1600            LDY #0
1610 .loop4
1620            LDA (start%),Y
                 \ Get the ASCII value
                of the first char in t
               he
1630            JSR &B50E
                 \ line. Use the routi
               ne at &B50E to PRINT it
1640            INY
1650            CPY linelen%
                 \ Finished the line y
               et ?
1660            BNE loop4
1670            JSR osnewl
                 \ PRINT a blank line
1680            RTS
1690            ]
1700   NEXT
1710 N%=0:FORK%=start TO P%-1:N%=N%+?K
       %:NEXT
1720 IFN%=40941 PRINT'" CHECKSUM corre
       ct"'"*SAVE FVAR ";~start" ";~P% :
       END
1730 PRINT'CHR$7"ERROR"'"Check the lis
       ting"
1740 END
1750
1760 DEFPROCtokens
1770 FORA%=128 TO 255
1780   IFA%>223 PRINTTAB(65,A%-224);:C
```

HANDBOOK OF PROCEDURES AND FUNCTIONS for the BBC Micro

- 143 pages
- Audrey and Owen Bishop ($27.95)

The first thing you notice about this book is the lack of an index. In it's place is a so-called "locator table" which purports to provide a cross-refernce of all the listed procedures and functions against their applications; quite a good idea but I think I would have preferred a detail index as well.

The book provides descriptions and listings of 70 procedures and functions, together with suggestions for enhancements. The description of each includes the following: What it does, Formal parameters, Local variables (if any), parameters, listing, how it works and how the program invoking it should operate.

Some of these procedures and functions are very useful and exhibit interesting and subtle programming techniques. Unfortunately some are obviously "fillers" and are of doubtful use. Examples of useful ones are: A routine to sort arrays (quicksort algorithm), A routines to display a character rotated 90 degrees, a routine to calculate a number to any base, a routine to scroll the screen downwards or sideways. Examples of less useful routines are: A routine to print a polygon at a specified point, a routine to delete a word from a linked list.

At nearly $30.00, this book is a bit pricey but on balance I would recommend it to anyone from novice to intermediate level to use as a BBC Basic programming reference.

* * *

```
      ALL&B50E:PRINTTAB(74,A%-224);A%
      ; ELSEIFA%>191 ANDA%<224 PRINTT
      AB(43,A%-192);:CALL&B50E:PRINTT
      AB(52,A%-192);A%;
1790   IFA%>159 ANDA%<192 PRINTTAB(22,
```

```
      A%-160);:CALL&B50E:PRINTTAB(30,
      A%-160);A%; ELSEIFA%<160 PRINTT
      AB(2,A%-128);:CALL&B50E:PRINTTA
      B(9,A%-128);A%;
1800   NEXT:G=GET
1810 ENDPROC
```

MEMORY DUMP

Steve C

Another program created out of necessity. This one displays the contents of memory on screen, eight bytes per line. Firstly it prints the current memory "pointer" value, then the contents of eight memory locations, the first of which is the contents of the location indicated by the pointer. On the same line, it then prints ASCII values of these values. If a value is equal to or greater than &7F, then an inverse "h" is printed, if a value is very low (<&20), it prints an inverse "l".

The psychology behind writing this program is not so much the "mother of invention" theory, but a "crisis inducment". The "invention" of the program is not significant - variations have been around for years, but I never got around to writing one to do just what I want until I had to try and find a program which I had written. I just couldn't remember where I had set PAGE! The program extended from there.

Operation is, I hope, simple. The user enters a start address. If a printout is required, s/he hits <RETURN> while holding down <COPY>, otherwise, s/he just hits return. If a printout is requested, the program prompts for a finish address. If this is entered, operation is automatic, outputting all memory location contents between the points entered. If 0 is entered, operation continues as normal, with a printout as well.

Once the start address is entered, the four arrow keys control operations. Up and down arrows move the display 8 bytes in either direction. The left and right arrows change the pointer value by one memory "page" (&100 bytes). When this is done, the new value is shown in inverse on-screen, otherwise the display is not affected. Page changes do not "auto-repeat" - I found it impossible to control accurately if it did repeat. To enter a new start address, hit <SHIFT> or <ESCAPE>. This also turns off the print option. To exit, hold <CTRL> while hitting <SHIFT>.

In order to allow this program to co-reside in memory with another, or with some data requiring checking, I have minimised the overhead memory space used. Around &80 bytes of memory is required between TOP and HIMEN. While it is possible to shorten this program to take up less than &300 bytes, inclulding this "running" space, by taking out REMs, spaces, and blank lines, as well as putting as many statements on one line as possible. However PLEASE type the whole thing in first. If you try to "save time" by shortening it as you go, I will get no credit for writing it, you will not be able to find typing errors, and will have no hope of understanding how the thing works. The BBC system is great for learning. If you don't care about the points I have noted here, you have wasted your money buying a quality computer, and should have bought a toy like an Atari or Sinclair!

Variables
W% - Z% - "double byte" values for VDU commands. (See your UG, (EUG p100))
O% - increment to memory pointer for each line of display
$S% - string used for converting input into HEX
P% - flag to indicate printing - 0 = FALSE, 2 = TRUE
F% - finish address - see above
I% - memory pointer itself
J% - counter

```
 10 REM Memory Dump Program
 20 REM By Steve Cosgrove
 30 REM 28 Sept, 1985
 40
 50 DIM S%5     : REM used to input st
    art address
 60 *fx4,2
 61 L% = 8 : REM Number of locations
    printed across the screen
 62 F% = 0 : REM used to print a cont
    inuous section of memory
 65 REM disable curser editing
 70 W% = &0011 : REM COLOUR  00
 80 X% = &8111 : REM COLOUR &81
 90 Y% = &0111 : REM COLOUR  01
100 Z% = &8011 : REM COLOUR &80
110 ON ERROR IF (ERR<>17) AND (ERR<>2
    8) THEN REPORT : PRINT " at line
    " ; ERL : OSCLI "FX4,0" : VDU3 :
    END : ELSE VDU 7
120 REPEAT
130   O%=0
135   CLS
140   INPUT "Start address "$S% : REM
       in HEX
150   P% = -2* INKEY(-106) : REM P% =
       2 if printout required, 0 othe
      rwise
155   REM P% is set if the COPY key i
      s held down while hitting RETUR
      N
160   I% = EVAL("&"+$S%)
170   $S%="0"
175   REM Get a Finish address if P%
      is set
180   IF P% THEN INPUT "Finish addres
      s "$S%
190   F% = EVAL("&"+$S%)
200   VDU P% : REM enable printer
210   REPEAT
220     I% = I%+O% : REM increment I%
         to desplay next line
230     PRINT RIGHT$("000"+STR$~(I% A
        ND &FFFF),4);"  ";
235     REM print I%, in HEX, with le
        ading zeros
236     REM if I% < 0 or I% > &FFFF,
        only output lower 4 places
240     FOR J% = I% TO I%+L%-1
250       PRINT RIGHT$("0"+STR$~(?J%)
          ,2);" ";
255       REM print contents of memor
          y location J%, with leading
           0
260       NEXT
270     VDU 32
290     FOR J% = I% TO I%+L%-1
300       IF (?J%>=&20) AND (?J%<127)
           THEN VDU ?J% ELSE IF ?J%>=
          127 THEN VDU 1,&E8,3,W%;X%;
          ASC("h"),Y%;Z%;P% ELSE VDU
          1,&EC,3,W%;X%;ASC("l"),Y%;Z
          %;P%
310       REM IF ?J% is printable, pr
          int it,
311       REM   ELSE IF its >= 127, p
          rint inverse "h"
312       REM            ELSE it must b
          e < 32, so print inverse "l
          "
313       REM &E8 & &EC print italic
          "h" and "l" on EPSON compat
          able printers
315       NEXT
320     REPEAT
330       O% = INKEY(-58)*-L%- INKEY(
          -42)*-L%- INKEY(-26)*-&100+
           INKEY(-122)*-&100
340       IF ABS(O%) > L% THEN I% = I
          %+O% : VDU3,W%;X%; : PRINT
          RIGHT$("000"+STR$~(I% AND &
          FFFF),4); : VDU 13,Y%;Z%;P%
           : REPEAT UNTIL NOT (INKEY(
          -26) OR INKEY(-122))
350       IF O% = L% THEN VDU 1,10,3,
          11,P%:REM scroll screen up,
           printer down
360       IF O% = -L% THEN VDU 10
           :    REM scroll screen up,
           printer down
365       IF (F% > I%) AND (F% > 0) T
          HEN O% = L% : VDU 10 ELSE F
          % = 0
```

```
370       VDU 13
380        UNTIL (ABS(O%) = L%) OR INK
         EY(-1)
390      UNTIL INKEY(-1)
```

```
400   VDU 1,10,3
410    UNTIL INKEY(-2)
420 CLS
430 *fx4,0
440 *fx15,1
```

THE PERFECT PROGRAMMER

"No program is that perfect."
     They said with a shrug.
"The client is happy ..
     What's one little bug?"

But he was determined.
     The others went home.
He dug out the flow chart
     Deserted, alone.

Night passed into morning.
     The room was cluttered
With memory dumps, microfiche,
     "I'm close," he muttered.

Chain smoking, cold coffee,
     Logic, deduction.
"I've got it," he cried. "Just
     change one instruction."

Then change two, then three more
     As year followed year.
And strangers would comment.
     "Is that guy still here?"

He died at the console
     Of hunger and thirst.
Next day he was buried
     Face down, nine edge first.

And his wife through her tears
     Accepted his fate.
Said, "He's not really gone,
     He's just working late."

From the Big Byte - courtesy PANSOPHIC

ENLARGED PRINTING

BILL T.

This program, BIGWORD, will display words and numbers on the printer in letters which are 7 lines high. Up to 10 characters can be displayed on each pass. The program runs on a Logitec dot-matrix FT-5000 printer, but should be adaptable for any printer capable of having user-defined characters.

OBJECTIVE

A need arose to have a program to run on an IBM PC which would print names from an existing file which would be of sufficient size to be used on name cards on conference tables. The BBC was selected on which to develop the program, as it was more convenient to do the development at home. This meant that the program should be independent of features specific to the BBC, thus precluding the use of the OSWORD call (A=&0A) which would have given an 8x8 matrix dot description rather than for each character having to be defined in DATA statements.

This independence was not taken to extremes - where a BBC statement had a direct equivalent in IBM BASIC then the BBC statement was used e.g. VDU is used to pass control codes to the printer, where the IBM eqivalent is LPRINT CHR$(n).

Ideally the program would have been printer independent, but the neccessity to download special characters prevented this, as the 2 printers concerned required different download actions. The program is clearly marked where changes need to be made for different printers.

PROGRAM FORMAT

The supplied program will produce big letters for any valid groups of up to 10 characters entered from the keyboard (only characters A-Z, 0-9 and SPACE are allowed in the version given here).

NULL input (just press RETURN) will terminate the program. This method should be used, rather than ESCAPE, in order to restore the printer & screen to initialised states.

The code required to actually produce the big letters is contained in procedures "biginit" and "bigprint". These procedures can be extracted and used in other programs, with "biginit" being called once before making any calls to "bigprint". Then "bigprint" can be called repeatedly, with the characters to be printed being passed as a string parameter to the procedure.

The program translates the string of input characters into a list of pointers which give the positions of each character in the array of character definitions. These pointers are used to obtain the portions of each character required to make up a line of print.

DEFINING THE LARGE CHARACTERS

DATA statements are used to define the format of the large characters. Each DATA statement consists of-

(a) the character being defined, for documentation reasons only

(b) 7 5-digit strings to define the character as 7 rows of 5 symbols. In each 5-digit string a zero (0) represents a "printed" space and one (1) represents an "inverse space" of a solid 7x5 block.

The DATA items are read into a 2-dimensional array in the "biginit" procedure. Currently the rows in the array are defined as follows -

- entry 0 = SPACE
- entry 1-26 = A - Z
- entry 27-36 = 0 - 9

Additional characters can be added to the end of the table, provided that the following lines are changed:-

Line 120 -
Amend N% to state the new table length

Line 1020 -
Amend the multiple nested IF statements to calculate table pointers from the ASCII codes of the added characters.

VARIOUS NOTES

*FX3,10 serves a dual purpose of directing PRINT output to the printer only and also turning the printer on (as with VDU2).

*FX3,0 restores output to the screen.

Beware of using *FX6,0 with this program (my printer is set to give automatic line-feed), because the zeroes will be suppressed while *FX3,10 is operative and even VDU1 cannot be used to force the zeroes through.

Any characters in the input string which are not defined to the program will be printed as spaces (large size!).

PROCEDURES

PROCbiginit -

line 120 - declare table size

line 130 - adjust for using entry 0

lines 150-200 - read data into table array

line 240 - determine required printing density

line 260 - direct output to the printer

lines 240-310 - initialise the printer, set emphasised printing if requested, and set 1/8" vertical spacing

lines 320-330 - download the print character definitions (called font registration on the Logitec) to replace symbols for 0 and 1

line 350 - return to screen display mode

PROCbigprint -

line 970-980 - calculate the TAB size to centralise the words on the printer

lines 990-1070 - translate each given character into a pointer into the print definition array and store the value of the pointer in a list of pointers for the current word

lines 1090-1160 - use the pointers to extract the data for each character on a line by line basis, and print it

PROCget -

requests the next set of characters to be printed

PROCfinish -

restores the printer to normal

PROCtitle -

sets up the screen display

VARIABLES

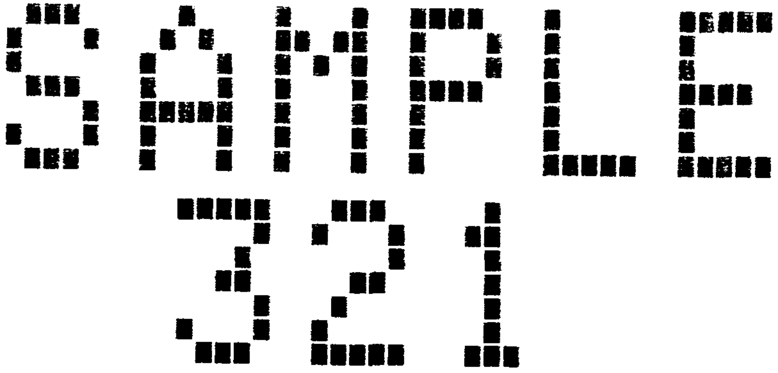
INTEGERS

A% - the length of the current input (max 10)
C% - the ASCII value of each character
I% - general purpose loop variable in FOR loops
J% - as for I%, in nested loops
N% - defines the size of the character definition array
tab% - the size of TAB required to centralise the printed line
ptr%() - array of pointers to the table positions of the characters to be printed

STRINGS

A$ - any string input from the keyboard
S$ - the string name used in PROCbigprint
L%() - the 2-dimensional array of character definitions as read from the DATA statements

---



```
10 REM ** T-BIGWORD **
20 REM
30 PROCtitle
40 PROCbiginit
50 PROCget
60 IF A$<>"" REPEAT PROCbigprint(A$)
     :PROCget:UNTIL A$=""
70 PROCfinish
80 END
90
100 DEFPROCbiginit
110 REM--------
120 N%=36:REM characs in table
130 N%=N%-1
140 REM define & load the table
150 DIM L$(N%,7)
160 FOR I%=0 TO N%
170   FOR J%=0 TO 7
180     READ L$(I%,J%)
190     NEXTJ%
200   NEXTI%
210 DIM ptr%(10)
220
230 REM set up the printer
240 PRINT"Single or double density (1
   /2)? ";
250 REPEAT:A$=GET$:UNTIL A$="1" OR A$
   ="2":PRINTA$
260 *FX3,10
270 REM------------------------------
   -------------------------
280 REM Adjust here for different pri
   nters
290 VDU27,64:REM INITIALISE
300 IF A$="2" VDU27,69:REM EMPHASISE
310 VDU27,48:REM 1/8" SPACING
320 VDU27,89,&30,&00,&00,&00,&00,&00,
   &00,&00,&00,&00
330 VDU27,89,&31,&FF,&FF,&FF,&FF,&FF,
   &FF,&FF,&FF,&FF
340 REM------------------------------
   -------------------------
350 *FX3,0
360
370 REM misc items
380 A$=STRING$(10,"X")
390 A$=""
400 VDU28,0,24,38,16
```

```
410 ENDPROC
420
430 REM Data to describe each charact
   er
440 REM
450 REM Space - also used for invalid
    characters
460 DATA " ",00000,00000,00000,00000,
   00000,00000,00000
470 REM
480 REM Letters A - Z
490 DATA A,00100,01010,10001,10001,11
   111,10001,10001
500 DATA B,11110,10001,10001,11110,10
   001,10001,11110
510 DATA C,01110,10001,10000,10000,10
   000,10001,01110
520 DATA D,11110,10001,10001,10001,10
   001,10001,11110
530 DATA E,11111,10000,10000,11110,10
   000,10000,11111
540 DATA F,11111,10000,10000,11110,10
   000,10000,10000
550 DATA G,01110,10001,10000,10000,10
   011,10001,01111
560 DATA H,10001,10001,10001,11111,10
   001,10001,10001
570 DATA I,01110,00100,00100,00100,00
   100,00100,01110
580 DATA J,00001,00001,00001,00001,00
   001,10001,01110
590 DATA K,10001,10010,10100,11000,10
   100,10010,10001
600 DATA L,10000,10000,10000,10000,10
   000,10000,11111
610 DATA M,10001,11011,10101,10001,10
   001,10001,10001
620 DATA N,10001,10001,11001,10101,10
   011,10001,10001
630 DATA O,01110,10001,10001,10001,10
   001,10001,01110
640 DATA P,11110,10001,10001,11110,10
   000,10000,10000
650 DATA Q,01110,10001,10001,10001,10
   101,10010,01101
660 DATA R,11110,10001,10001,11110,10
   100,10010,10001
670 DATA S,01110,10001,10000,01110,00
   001,10001,01110
680 DATA T,11111,00100,00100,00100,00
   100,00100,00100
```

```
690 DATA U,10001,10001,10001,10001,10
   001,10001,01110
700 DATA V,10001,10001,10001,01010,01
   010,00100,00100
710 DATA W,10001,10001,10001,10101,10
   101,10101,01010
720 DATA X,10001,10001,01010,00100,01
   010,10001,10001
730 DATA Y,10001,10001,01010,00100,00
   100,00100,00100
740 DATA Z,11111,00001,00010,00100,01
   000,10000,11111
750 REM
760 REM Numbers 0 - 9
770 DATA 0,00100,01010,10001,10001,10
   001,01010,00100
780 DATA 1,00100,01100,00100,00100,00
   100,00100,01110
790 DATA 2,01110,10001,00001,00110,01
   000,10000,11111
800 DATA 3,11111,00001,00010,00110,00
   001,10001,01110
810 DATA 4,00010,00110,01010,10010,11
   111,00010,00010
820 DATA 5,11111,10000,11110,00001,00
   001,10001,01110
830 DATA 6,00110,01000,10000,11110,10
   001,10001,01110
840 DATA 7,11111,00001,00010,00100,01
   000,01000,01000
850 DATA 8,01110,10001,10001,01110,10
   001,10001,01110
860 DATA 9,01110,10001,10001,01111,00
   001,00010,01100
870
880 DEFPROCget
890 REM-------
900 INPUT "Enter up to 10 characters
   "A$
910 IF LEN(A$)>10 A$=LEFT$(A$,10)
920 ENDPROC
930
940 DEFPROCbigprint(S$)
950 REM-----------
960 LOCAL A%,C%,I%,J%
970 A%=LEN(S$)
980 tab%=(10-A%)/2 *6 +6
990 FOR I%=1 TO A%
```

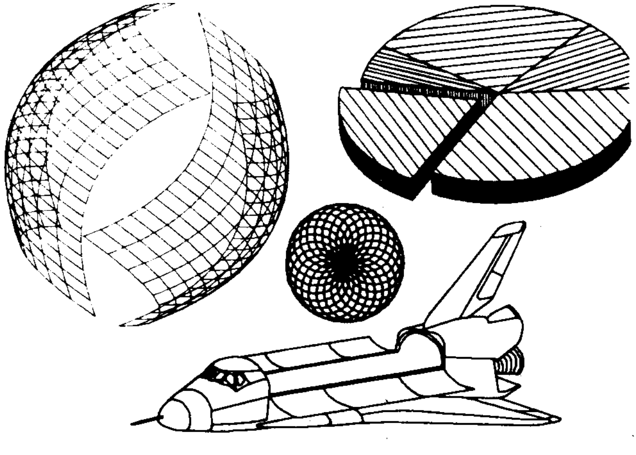
```
1000   C%=ASC(MID$(S$,I%,1))
1010   REM Translate to table index
1020   IF C%>64 AND C%<91 C%=C%-64 ELS
   E IF C%>47 AND C%<58 C%=C%-21 E
   LSE IF C%=32 C%=0 ELSE C%=0
1030
1040   REM store each index
1050   ptr%(I%)=C%
1060   NEXTI%
1070
1080 REM Now print it
1090 *FX3,10
1100 FOR J%=1 TO 7
1110   PRINTTAB(tab%);
1120   FOR I%=1 TO A%
1130     PRINTL$(ptr%(I%),J%);"  ";
1140     NEXT I%
1150   PRINT
1160   NEXT J%
1170 PRINT
1180 *FX3,0
1190 ENDPROC
1200
1210 DEFPROCfinish
1220 REM----------
1230 VDU2
1240 VDU1,27,1,64:REM RESET PRINTER
1250 VDU3
1260 VDU26
1270 CLS
1280 PRINT"BYE"
1290 ENDPROC
1300 DEFPROCtitle
1310 REM---------
1320 CLS
1330 PRINTTAB(10,0);CHR$(129);CHR$(141
   );"BIGWORD"
1340 PRINTTAB(10,1);CHR$(129);CHR$(141
   );"BIGWORD"
1350 PRINTTAB(0,4);"Prints up to 10 ch
   aracters at a time"'"in enlarged
   letters"
1360 PRINTTAB(0,7);"Allowable characte
   rs are:"TAB(4,9);CHR$(130);"A-Z,
   SPACE, 0-9"
1370 PRINTTAB(6,11);"(Invalid characte
   rs are"'TAB(7);"printed as spaces
   )"'
1380 ENDPROC
```

Space Quest

JL

This is a game in Mode 7. It was written originally to directly access the screen with ? and $ the indirection operators. I since realised that this deprives a substantial portion of the membership of the chance to use it. It has been changed to use only approved methods such as vdu's print tabs etc.. It relies on the fact that control codes for colours etc. are not visible and cannot be seen as different from spaces which are also the background colour black. Mode 7 graphics are limited but colourful.

After a Title and Introduction the screen is filled with solid lines of colour which are then broken up by random control codes and spaces. A 'ship' is moved about the screen with the cursor keys and the object landed on is determined by FNchr, (U.G.p.432 and others) with the action taken determined by Procgo. Various messages are displayed from time to time depending on the current situation. Movement with the cursor keys is only horizontal or vertical but landing on control codes causes movement diagonally towards the centre of the screen. Any positions landed on are changed to empty space i.e. vdu 32. I hope that the variables and Procnames are self explanatory, changing them could lessen the typing. The program contains some anomalies; no prize for spotting them but instant fame for showing how to overcome them. Happy R.S.I..



```
10 REM ** SPACE QUEST **
20 REM **   JL    85  **
30 REM **   MODE   7  **
40 MODE 7
50 ENVELOPE 1,1,0,0,0,0,0,0,126,-1,
   -1,-15,126,0
60 ENVELOPE 2,1,0,0,0,0,0,0,50,0,0,
   -50,126,126
70 ENVELOPE 3,1,5,0,0,50,0,126,126,
   0,-10,126,126,0
80 ENVELOPE 4,1,130,0,0,33,0,0,1,-1
   ,0,0,124,10
90 ENVELOPE 5,1,17,0,0,58,0,0,1,-1,
   0,0,7,7
100 ENVELOPE 6,1,55,0,0,14,0,0,14,0,
    0,0,38,0
110 ENVELOPE 7,2,68,0,0,33,0,0,1,-1,
    0,0,124,8
120 CLS
130 VDU 23;8202;0;0;0;
140 PROCtitle
150 PROCintro
160 tough%=20
170
180
190 REPEAT
200   CLS
210   middleage=39
220   full%=25
230   cargo%=25
240   before%=49
250   SOUND 3,7,0,100
260   PROCopening
270
280
290   REPEAT
300     SOUND 3,5,0,5
310     PRINT TAB(15,22)SPC(6)"DRIFT
        ING"SPC(6)
320     VDU 31,15,24,32,85,83,69,32,
        32,67,85,82,83,79,82,32,32,7
        5,69,89,83,32,32,32,32
330     PROCmove
340     UNTIL middleage = 40
350   UNTIL middleage = 41
360 END
370
380
390 DEF PROCmove
400 *FX 21,0
410 REPEAT
420   F%=GET
430   UNTIL F%>0
440 IF F%=84 F%=0:PROCtransmogrify:E
    NDPROC
450 VDU 31,side%,up%,32
460 IF F%=136 AND side%>1 side%=side
    %-1:PROCgo:ENDPROC
470 IF F%=137 AND side%<39 side%=sid
    e%+1:PROCgo:ENDPROC
480 IF F%=138 AND up%<19 up%=up%+1:P
    ROCgo:ENDPROC
490 IF F%=139 AND up%>1 up%=up%-1:PR
    OCgo:ENDPROC
500 VDU 31,side%,up%,35
510 ENDPROC
520
530
540 DEF PROCgo
550 round%=FALSE
560 REPEAT
570   VDU 31,side%,up%
580   what%=(FNchr(&87,0,0) AND &FF0
      0) DIV &100
590   IF what% >48 AND what% <53 PRO
      Cgalaxy:round%=TRUE
600   IF what% = 255 VDU 31,side%,up
      %,35:cargo%=cargo%+1:PROCbooty
      :full%=full%+1:PROCfuel:round%
      =TRUE
610   IF what% = 32 full%=full%-1:PR
      OCfuel:round%=TRUE
620   IF what% >144 AND what% <151 V
      DU 31,side%,up%,32:PROCblackho
      le
630   IF what% = 129 PROCedge:round%
      =TRUE
640   UNTIL round%=TRUE
650 VDU 31,side%,up%,35
660 ENDPROC
670
```

```
680
690 DEF FNchr(A%,X%,Y%) = USR&FFF4
700
710
720 DEF PROCfuel
730 IF full%<15 PROCdisaster:ENDPROC
740 IF full%>=39 full%=39
750 PRINT TAB(15,20)SPC(25)
760 FOR zip=15 TO full%
770   VDU 31,zip,20,255
780   NEXT
790 ENDPROC
800
810
820 DEF PROCbooty
830 SOUND 1,-8,200,1
840 IF cargo% <15 VDU 31,15,1,32:END
    PROC
850 IF cargo%>=39 cargo%=39
860 PRINT TAB(15,1)SPC(25)
870 FOR zip=15 TO cargo%
880   VDU 31,zip,1,255
890   NEXT
900 ENDPROC
910
920
930 DEF PROCblackhole
940 SOUND 1,3,5,10
950 SOUND 1,3,66,7
960 PRINT TAB(15,22)SPC(5)"BLACK   H
    OLE"
970 VDU 31,15,24,68,82,73,86,69,32,3
    2,68,82,73,86,69,32,32,68,82,73,
    86,69
980 IF up% <10 U% = 1 ELSE U% =-1
990 IF side% <20 S% = 1 ELSE S% =-1
1000 up%=up%+U%:side%=side%+S%
1010 full%=full%-1
1020 IF full%<15 round%=TRUE:PROCdisa
     ster:ENDPROC
1030 PROCfuel
1040 FOR taihoa=1 TO 1000
1050   NEXT
1060 ENDPROC
1070
1080
1090 DEF PROCedge
```

```
1100 IF up%<10 THEN up%=up%+1 ELSE up
     %=up%-1
1110 ENDPROC
1120
1130
1140 DEF PROCgalaxy
1150 which%=what%-48
1160 IF before%-what%<>0 PROCwrongord
     er:ENDPROC
1170 IF which%=4 PROChome:ENDPROC
1180 PRINT TAB(15,22)"  GREAT  GOING
     "
1190 VDU 31,15,24,32,32,32,82,69,65,6
     8,89,32,84,79,32,68,69,80,65,82,
     84
1200 FOR times%=1 TO which%
1210   PROCbells
1220   NEXT
1230 before%=before%+1
1240 full%=39:PROCfuel
1250 cargo%=15+2*(cargo%-15):PROCboot
     y
1260 FOR taihoa = 1 TO 1300*which%
1270   NEXT
1280 IF which%<3 THEN up%=up%+1 ELSE
     up%=up%-1
 1290 VDU 31,side%,up%,35
1300 ENDPROC
1310
1320
1330 DEF PROCwrongorder
1340 PRINT TAB(15,22)"  WRONG   ORDE
     R !!  "
1350 FOR taihoa =1 TO 2000
1360   NEXT
1370 IF which%<3 THEN up%=up%+1 ELSE
     up%=up%-1
1380 VDU 31,side%,up%,35
1390 ENDPROC
1400
1410
1420 DEF PROChome
1430 SOUND 3,0,0,0
1440 PROCbells:PROCbells
1450 value=1000*tough%*(cargo%-15)
1460 FOR zip=15 TO 38
1470   VDU 31,zip,22,32
```

```
1480   VDU 31,zip,24,32
1490   NEXT
1500 PRINT TAB(15,22)SPC(8)"OUR   HER
     O"
1510 VDU 31,20,24,32,32,87,69,76,67,7
     9,77,69,32,32,72,79,77,69
1520 FOR zip=0 TO 39
1530   VDU 31,zip,0,32
1540   VDU 31,zip,1,32
1550   NEXT
1560 VDU 31,0,0,129,141
1570 VDU 31,0,1,129,141
1580 PRINT TAB(3,0)"Your reward is  $
      ";value
1590 PRINT TAB(3,1)"Your reward is  $
      ";value
1600 FOR taihoa = 1 TO 4000
1610   NEXT
1620 PRINT TAB(15,22)"  MORE GALAXIES
      AWAIT"
1630 VDU 31,16,24,70,65,83,84,69,70,3
     2,89,79,85,82,32,83,80,65,67,69,
      66,69,76,84
1640 FOR A% = 139 TO 1 STEP -3
1650   SOUND 1,-12,A%,1
1660   NEXT
1670 FOR taihoa = 1 TO 2500
1680   NEXT
1690 middleage=40
1700 tough%=tough%+5
1710 ENDPROC
1720
1730
1740 DEF PROCtransmogrify
1750 PRINT TAB(15,22)SPC(25)
1760 PRINT TAB(15,22)" T  TO  TRANSFE
     R "
1770 VDU 31,15,24,32,83,80,65,67,69,3
     2,32,66,65,82,32,32,69,78,68,83,
     32,32
1780 REPEAT
1790   SOUND 3,6,0,10
1800   key = GET
1810   IF key = 84 AND full%<39 PROCc
       hange
1820   *FX15,0
1830   UNTIL key = 32
```

```
1840 ENDPROC
1850
1860
1870 DEF PROCchange
1880 IF cargo%<15 ENDPROC
1890 FOR A%=1 TO 100 STEP 6
1900   SOUND 1,-12,A%,1
1910   NEXT
1920 IF full%>=39 ENDPROC
1930 cargo%=cargo%-1:full%=full%+1
1940 PROCbooty:PROCfuel
1950 ENDPROC
1960
1970
1980 DEF PROCdisaster
1990 SOUND 3,0,0,0
2000 PRINT TAB(15,20)" DRIFTING HELPL
     ESSLY"
2010 PRINT TAB(15,22)"  ANOTHER TRY
     ?      "
2020 VDU 31,15,24,32,32,32,80,82,69,8
     3,83,32,32,89,32,32,32,32,32,32,
     32,32
2030 VDU 31,15,20,32
2040 FOR A% = 139 TO 1 STEP -3
2050   SOUND 1,-12,A%,1
2060   NEXT
2070 REPEAT
2080   key=GET
2090   UNTIL key = 89 OR key =121
2100 middleage=40
2110 tough%=tough%-5
2120 ENDPROC
2130
2140
2150 DEF PROCbells
2160 SOUND 1,1,131,10
2170 SOUND 2,2,131,20
2180 SOUND 1,1,131,10
2190 ENDPROC
2200
2210
2220 DEF PROCopening
2230 PRINT TAB(0,20)CHR$(134)SPC(6)"U
     SE CURSOR KEYS TO MOVE"
2240 PRINT TAB(0,22)CHR$(134)SPC(9)"W
     ATCH THE SCREEN"
```

```
2250 FOR row = 3 TO 18
2260   zip=144+RND(6)
2270   VDU 31,0,row,zip
2280   NEXT
2290 FOR column = 1 TO 39
2300   FOR row = 3 TO 18
2310     zip = 255
2320     VDU 31,column,row,zip
2330     NEXT row:NEXT column
2340 PRINT TAB(5,20)"SUCCESS GIVES A
     HARDER MISSION"
2350 PRINT TAB(6,22)"FAILURE GIVES AN
      EASIER TASK"
2360 FOR row = 3 TO 18
2370   FOR hard= 1 TO tough%
2380     column = RND(39):zip = 144+R
         ND(6)
2390     IF column = 0 column = 1
2400     VDU 31,column,row,zip
2410     NEXT hard:NEXT row
2420 FOR row = 3 TO 18
2430   FOR column = 1 TO 39
2440     space = RND(3)
2450     IF space = 2 VDU 31,column,r
         ow,32
2460     NEXT column:NEXT row
2470 FOR zip = 1 TO 39
2480   VDU 31,zip,2,129
2490   VDU 31,zip,19,129
2500   NEXT zip
2510 VDU 31,0,2,136,9,49,31,38,2,50
2520 VDU 31,0,19,136,9,51,31,38,19,52
2530 VDU 31,0,20,129,157,132,32,70,85
     ,69,76,32,32,32,156,137,140,145
2540 VDU 31,0,22,130,157,132,83,84,65
     ,84,85,83,32,32,156,137,140,130
2550 VDU 31,0,24,131,157,132,65,67,84
     ,73,79,78,32,32,156,137,140,131
2560 VDU 31,0,1,133,157,132,67,65,82,
     71,79,32,32,156,137,140,154,149
2570 REM all prints to start at (15,y
     )
2580 side%=20:up%=18
2590 VDU 31,side%,up%,35
2600 *FX 4,1
2610 PROCfuel
2620 PROCbooty
```

```
2630 ENDPROC
2640
2650
2660 DEF PROCtitle
2670 PRINT TAB(10,10)CHR$(129)CHR$(14
     1)"SPACE  QUEST"
2680 PRINT TAB(10,11)CHR$(129)CHR$(14
     1)"SPACE  QUEST"
2690 FOR times = 1 TO 2
2700   PROCbells
2710   NEXT
2720 FOR taihoa = 1 TO 4000:NEXT
2730 CLS
2740 ENDPROC
2750
2760
2770 DEF PROCintro
2780 SOUND 3,4,0,48
2790 PRINT CHR$(131)SPC(5)" YOU have
     been CHOSEN to"
2800 PRINT'CHR$(131)SPC(4)"COMMAND  S
     TARSHIP HASHMAK"
2810 PRINT'CHR$(131)SPC(2)"on its voy
     age around the universe"
2820 PRINT CHR$(131)"to restore the f
     ortunes of your world"
2830 PRINT'CHR$(129)SPC(2)"On your tr
     avels landing on coloured"
2840 PRINT CHR$(129)SPC(2)"planets in
     creases your possessions"
2850 PRINT'CHR$(131)SPC(2)"Empty spac
     e may carry no penalty but"
2860 PRINT CHR$(131)SPC(5)"BLACKHOLES
      consume fuel"
2870 PRINT CHR$(131)SPC(4)"and ruin y
     our navigation."
2880 PRINT'CHR$(129)SPC(2)"Visit each
      numbered galaxy in turn"
2890 PRINTCHR$(129)SPC(3)"topping up
     your fuel tank and"
2900 PRINT CHR$(129)" doubling your c
     argo although both may"
2910 PRINT CHR$(129)SPC(9)"never exce
     ed full."
2920 PRINT'CHR$(131)"To convert cargo
      to fuel   Press .. T"
2930 PRINT CHR$(131)SPC(4)"Use the cu
```

```
     rsor keys to move"
2940 PRINT CHR$(129)SPC(3)"Press the
     space bar to begin"
2950 PRINT'CHR$(131)SPC(9)"WATCH  THE
     SCREEN"
2960 REPEAT
2970   key=GET
2980   UNTIL key = 32
2990 ENDPROC
```

RTTY

Ted M ZL1BOY Arapuni

Including a review of G3WHO's RTTY program.

Perhaps before starting a review, some explanation of what RTTY is all about is in order. RTTY stands for Radio TeleTYpe, and such is the trademark of the Teletype Corp of the USA. A better name for it is Radio Tele-printing. The code used for this method of communication is known as the BAUDOT code, or as the Murray Multiplex code, and is vaguely similar to the more familiar ASCII code. However the major difference is in its format: ASCII uses 8 bits to describe each letter, whereas Baudot has only 5 bits. It will be noted that there are only 32 possible combinations of 5 bits, and in fact only 31 are used. As there are 26 letters in the alphabet, there are insufficient combinations, so the solution is to use the code twice, placing the alphabet and machine functions on it the first time round, sending a special code to tell the machine to "change state" and then sending the code all over again with numbers and punctuation in it. All sounds very complicated, but the system works, and has done so for a very long time - in fact it was dreamed up in 1874! The code is sent in serial form, with the addition of 1 start bit and 1 1/2 stop bits, giving 7 1/2 bits per 'byte'.

This system has been used for years by the Post Office and like organisations around the world for telegraph traffic and is well known in this country as the TELEX. It is also used extensively on the shortwave radio bands by all sorts of organisations for news bulletins, weather broadcasts, international telex, military information to name a few. It is also used by some radio amateurs (myself included) as a means of communication, instead of phone or cw (morse). Its chief drawback is that it is fairly slow - usually 50! or 75! bauds, amateur standard being 45.45(!!) bauds, just to be different.

Up until the advent of computers, anyone who wanted to 'print' RTTY had to acquire a teleprinter such as the Creed 7B, 54, or 75, and these were not easy to get until recent times. They are also noisy, messy things that don't endear themselves to the rest of the household! The micro changed all that and made possible a very sophisticated rtty setup. I have been dabbling with the mode for the past 10 years, with mechanical printers, and the acquiring of a Beeb with printer made me want to find suitable rtty software. I found what I wanted in an English radio magazine, and I sent away to Peter for more info. It appeared this software was available on cassette for &7.30 or disc &9.50 or ROM &20 The ROM appealed to me so off went the money and back came the ROM 3 weeks later without any Customs problems.

I should make it clear at this stage, anyone who wishes to use this mode of communication will require as well as the Beeb, a good shortwave receiver and a modem to convert the audio tones from the receiver into TTL-compatible dc pulses, much the same as a telephone modem. Suitable receiving converters are published in the amateur radio press from time to time, and I would be happy to help anyone out with suitable circuitry. If you wish to transmit teletype you will require an amateur's transmitting licence as it is against the law to transmit on amateur frequencies without one.

Anyway, having installed the ROM it was powered up and set going. Presto! Up came the command to insert my callsign, then set the internal 24-hour clock. The program is menu driven so there are a range of special functions to access, such as setting different speeds, programming one of the 6 255 character transmit memories, the contents of which can be saved to tape or disc to save reprogramming them each time they are used.

When either of the two operating modes are selected (50 & 45.45 baud), the computer then displays in split screen mode. This means that received text is displayed in the top half of the screen, and text prepared for transmission is displayed on the bottom half. The program uses Mode 3, so 80 characters per line are displayed. Along the top of the display is a line containing: the status of the program ie receive or transmit, USOS enabled, printer on or off, and the contact number. USOS is a handy feature for rtty as, due to atmospheric conditions, the command to the recieving printer to return to "letters" shift after printing figures is sometimes mutilated and the printer stays locked in "figures" shift. USOS (UnShift On Space) means that the computer goes back to letters shift on receipt of a valid 'space' character. There is also a useful callsign search facility that searches received text for a callsign and puts into memory, so when transmit is selected, your call and the call of the station you are working is sent at the beginning of the transmission. The text for transmission is written in the lower half of the screen and text may be prepared while the unit is also receiving text in the upper half. Another advantage is that all memory contents that are transmitted are also displayed on the screen so you can see exactly what is going on. The transmit screen has two cursors, one controlled from the keyboard, and one that displays the character that is being transmitted, allowing you to type ahead of the transmit cursor.

As well as all this, the receive and transmit speeds may be altered from the keyboard in 5 baud steps between 10 and 100 baud, to allow the reception of non-standard speeds. These are quite common on shortwave bands. There is also a real time clock that can be inserted into transmitted text which adds a professional touch. The ROM version has the facility to allow the Beeb to be used to generate morse code for transmission. This is a nice touch, and is far better to use than the old morse key! Any printer that is normally used with the Beeb may be used with the program; when the printer is enabled, each line is printed out from both received and transmitted text at the end of each line of the display.

What do I think of it? Magic!! Over the years I have seen a lot of RTTY software for various computers; I have owned both an Ohio Superboard 2 and a 16K ZX81 and had rtty for both of them; I have seen other computers with rtty fitted; none have compared with this one. I make no apology about my love affair with the Beeb - my wife doesn't see it that way I'm sure - and this piece of firmware does everything that an avid RTTY'er could want. Having said that, it would be nice to see ASCII included with the package, but there isn't much of it about on the airwaves yet, and, in my opinion, it is not a code that lends itself very well to radio propagation, for a number of reasons I won't go into here. I consider that this soft/firmware is well worth the money.

Where do you get it?
Mr P.J. Harris G3WHO, 10 Appleby Close, Great Aline, Alcester, Warwickshire B49 6HJ UK
Peter also has other software he has wriiten for sale, including AMTOR which is a funny sort of teletype that is becoming popular with the amateur fraternity.

AMATEUR RADIO SOFTWARE

Reviewer: Brian C ZL2BFC Wgtn

1. Slow scan television (G8UEE / G3LIV)
2. CW QSO (Scarab)
3. Morse Tutor (Scarab)
4. RTTY (G3WHO)
5. RTTY (Scarab)

1 SLOW SCAN

This software is designed to turn your BBC computer into a monitor for SSTV. The package consists of a cassette with software on one side and some test pictures on the other. My copy was well recorded and loads reliably each time. There is also a PCB and instructions for the hardware decoder required. The PCB is of good quality, and predrilled. Construction is straightforward (Knowledge of electronics required). You will need to buy the components to fit on it. Alignment can be done with a multimeter using the test patterns on the cassette.

To use the package, audio from the receiver is fed to the decoder and a ribbon cable connects to the Beeb, but in my case a separate power supply is used.

Upon loading the program and connecting a suitable source of audio to the input, the screen will come alive with the incoming picture. Various options allow variable width, positive or negative imaging, frame retrigger (clears screen also), and choice of 7, 16, 32 second formats. It will handle American 60 Hertz video as well.

As for criticism - I have none. It is an excellent unit and is fun to use.

Source: S.P. Electronics 48 Linby Rd, Hucknall, Notts NG15 7TS, Great Britain.
Cost: about $45.

2 CW QSO

A full featured Basic program with machine code that will enable 2-Way communication on Morse code using the computer to decode and display in English the received signal. It sends Morse after it has been typed in English to the keyboard too. The package has a cassette and a manual fully describing the operation of the program and the interconnection details etc. By the way, I should make it clear that an external encoder/decoder unit is needed for each mode. The slow scan does not need an encoder however as it does not transmit.

Upon running the program a message appears asking for speed. If the figure shown is oK, hit return else enter your own figure. This will set the transmit speed and tell the receiver about what speed to expect to decode. The receive speed automatically adjusts to the incoming signal but leaves the transmit side alone. From then on any signals once tuned in will print straight onto the screen until the transmit mode is entered. Any words then typed in will then be sent out as Morse. A sidetone tells you what is being sent. The red keys can be programmed for short messages and a printer can be used for hard copy.

My only criticism of this extremely versatile program is that although it has a keyboard buffer allowing you to type ahead, the text in that buffer is hidden from view - too bad if you make a mistake in your typing. Apart from that one criticism, again full marks.

Source: Scarab Systems, 39 Stafford St, Gillingham, Kent ME7 5EN UK.
Cost: round $25.

3 MORSE TUTOR

I wish I had had this program when I was learning my morse. It is excellent in all respects as a tutor. Package is cassette and instructions. It is a very long program and will not fit neatly in memory with discs.

Upon running it, the morse alphabet is displayed on screen to save you trouble looking up the codes. A whole series of questions follows with default values asking about speed, weightings, text etc. A loud burst of tone out of the loudspeaker and a suitable message on screen tell you to get ready. All very dramatic. The morse then comes from the loudspeaker and at the end a prompt asks for the next step. The program is user-friendly. One of the nice features is the lookup table containing 750 plain English words to read from. The chances of guessing the word are fairly remote.

Another surprising element of this program is that it will test your sending as well. By hooking up a key to one of the inputs on the User port it will decode your morse and tell you what speed it was sent at. Any errors will undoubtedly be your own doing. Alternately, a decoder hooked up will enable off-air signals to be decoded as well as indicating the speed. This speed indicator is not present in the 2-way program above.

All in all a thoroughly useful program.
Source: Scarab
Cost: about $15.

4 RTTY (G3WHO)

A 2-way teletype program using suitable interfaces. This is my favourite mode and the program makes it worth while. There are so many features available that I prefer it to the one detailed below. The first thing that impressed me was the split screen arrangement enabling received signals to appear in the top half of the screen, and your reply to be entered in the lower part. You can type in while the other station is sending to you. There is an auto callsign search which will look for the other stations callsign and poke it into a memory for you to recall at pleasure. The red keys can be programmed with messages of your own creation and saved as part of the program proper. Other features include morse code id (if desired), real time id, adjustable speed, printer hookup etc.

The manual is again up to standard and is comprehensive. The program is also available on Eprom, but from what I have heard the cassette version is better.

Source: S.P. Electronics.
Cost: about $25

5 RTTY (Scarab)

A 2-way program also but it differs in that a conventional screen arrangement is used, thus it is not possible to type ahead while in Receive mode. It also has the curious habit of sending the morse id via the Beeb's loudspeaker and not to the transmitter if a separate encoder is used. Also when you run the program, GMT and day etc are asked for, but whereas the time is updated automatically, the date isn't. Should you go past midnight GMT (noon to us Kiwis) the date will need to be manually incremented.

For all that though, it does have one trick up its sleeve, in that you can have 40 or 80 character widths with corresponding definition and print size. The 40-character size is good if your monitor is not up to scratch. The manual is as good as the others.

Source: Scarab
Cost: about $30

INTERFACING

It is a credit to all programmers (except the SSTV which has different requirements) that all the pin connections to the Beeb are done the same way for each program. eg PB2 is the pit line and PB0 is the the receive input. Even the TTL pulses are the same way up. Interfacing is a breeze because of that.

In my experience it is vital to have good decoders on the receive side, whereas the encoders are not so critical. Included with the CW QSO program is a circuit for a simple decoder using a single NE567 IC. Building that decoder is a waste of time because the waveform coming out of it is not clean enough for the Beeb to handle. Only garbage will appear on screen. Instead I would suggest building the circuits out of 'Elektor' in the series published during 1983 on amateur radio. For encoding I have a simple design for morse and also for RTTY that I am prepared to pass on (see below).

CONCLUSION

I have outlined 5 of the software packages available to the amateur as I have experienced them. There are others available, but I have no information on them. Scarab seem to be the leaders in this field and they have a large range of software for most of the computers around.

All the programs were supplied on cassette with no protection and thus are easy to transfer to disc. This was done deliberately by the programmers to enable you to put in messages etc and make backup copies. The ability to pirate is a risk they took and the manuals make it clear how they feel on that subject.

If anyone would like copies of circuits (but NOT the software itself), drop me a line and $5 for photocopying etc:
Brian Cattermole, 34 Routley Cres, Upper Hutt. I can be found monitoring 3883 Khz most evenings at 2100 hours.

A PRINTER?

Anne F.

So you're tossing up about whether you need a printer or not. There are some obvious uses:

1. Print out listings when your programs are too big to fit on the screen all at once.
2. Do all your correspondence.
3. (if it's an old printer or teletype machine) put it out in the orchard as a bird scarer.

But there is more to it than this.
How about buying a drawing package as well, or using Bill T's recent one, and making your own pictures. These could provide you with a lifetime supply of unique and personalised Christmas cards, Birthday cards, Wedding invites!

Daisy (nee 32-0-16) and BoB Comp
are proud to announce
a new peripheral to their family...

and so on. Your printer can take coloured paper, light card for this.

Pictures may be put in a scrap book or travelling book - a stylised geyser for the kids to spot as you drive through Te Aroha or Rotorua, or a trig station (who sees the most)... and the kids will have had some anticipatory days of fun designing their computer pics to put in the book while you get under the car, pack suitcases etc.

If you have a penchant for adventure games, do a CTRL-B when you start, and you will get a printout of the game and your responses for future reference. This chews through the paper though.

You may know someone with poor eyesight, who still likes to receive letters. Offered here is a Basic program which will take a letter written in Wordwise, space it over how many screens are needed, and then dump the text. The screen dump routine here for a C.Itoh produces a A4-wide page of text. This results in much larger, darker text for reading. It doesn't matter that a screen has up to 25 lines, and your letter is 5 pages long! The program prompts for how many lines you want on each screenfull (22 lines looks good). It then gets the first (eg 22) lines of text from the Wordwise file, spaces them evenly down the screen, then calls the screen dump routine. When the dump is finished, it clears the screen, gets the next 22 lines... and so on until the end of your letter is reached.

Preparing the text file:
Within Wordwise, write your letter.
Set LL40
Save using option 8 from the menu.
Back in to Basic

```
   10 REM  TEX_DMP
   20 REM To read a spooled WORDWISE text,
   30 REM space it evenly on the screen
   40 REM then to a double-sized screendump
   50 REM for each screen.
   60 REM by Gavin F 3/3/85
   70 REPEAT
   80   INPUT "Spooled text file to print is ? "f_text$
   90   UNTIL LEN(f_text$) > 0
  100 c_text% = OPENIN(f_text$)
  110 PRINT''
  120 REPEAT
  130   INPUT "Number of lines per screen [8...25] ? "M%
  140   UNTIL M%>7 AND M%<26
  150 MODE4
  160 L%=1
  170 y_inc=1024/M%
  180 MOVE0,1023
  190 VDU5
  200 REPEAT
  210   X% = BGET#c_text%
  220   IF X%=&0D THEN MOVE0,1023-(L%*y_inc):L%=L%+1
  230   IF X%>&1F AND X%<&7F THEN VDUX%
  240   IF L%>M% THEN PROC_screen_dump:CLS:L%=1
  250   UNTIL EOF#c_text%
  260 CLOSE#c_text%
  270 IF L%>1 THEN PROC_screen_dump
  280 VDU4
  290 END
  300
  310 DEF PROC_screen_dump
  320 LOCAL A%,B%,C%,D%,E%,F%,G%,X%,Y%,Z%
  330 A%=1023:B%=15:C%=-16
  340 D%=0:E%=1279:F%=4
  350 G%=0
  360 MOVE1279,1023
  370 DRAW1279,1022
  380 VDU2,1,24,1,27,1,84,1,49,1,54,1,2,1,27,1,62
  390 FOR Y%=A% TO B% STEP C%
  400   VDU1,27,1,83,1,48,1,54,1,52,1,48
  410   FOR X%=D% TO E% STEP F%
```

```
 420    Z%=0
 430    IF POINT(X%,Y%-12)<>G% THEN Z%=&C0
 440    IF POINT(X%,Y%-8)<>G% THEN Z%=Z% OR &30
 450    IF POINT(X%,Y%-4)<>G% THEN Z%=Z% OR &0C
 460    IF POINT(X%,Y%)<>G% THEN Z%=Z% OR &03
 470    VDU1,Z%,1,Z%
 480    NEXT
 490   VDU1,13
 500   NEXT
 510 VDU3,1,27,1,69,1,27,1,65,1,27,1,60,3
 520 ENDPROC
```

IS IT EXPENSIVE?

Like everything, that depends...

1. Printer ribbon - carbon once-only is expensive, nylon ribbons go round and round gradually fading.

Some may be satisfactorily re-inked, and a ribbon cartridge is easily opened up for this - just do it carefully so metres of bunched-up ribbon doesn't fall out. What ink is important. Tony W has been researching this, along with ribbon replacements for Logitecs, and finds there is a special dot-matrix ink, which is reduced in pigment that should be used. With other inks there is no guarantee that the pigment won't make the pins on the printer head stick or jam or wear down quicker.

The life of a ribbon depends in part on its length, and in part, on what you do to it. Lots of screen dumps wear ribbons out pretty fast. If it is practical for you, have two ribbons - one newer one for text, an older one for graphics.

2. Paper - from $0 to $lots, it's your choice.

$0 even though you probably have an A4-width printer, the wide commercial printout paper can be sliced to an appropriate width, then used on friction feed - don't try to use the sprocket holes on one side.

$0 some word-processing paper sheets are backed on to sprocketted fanfold paper. This backing paper is lovely quality, the only disadvantage for some being that what I have seen is perforated every half page.

$5+ packets (reams) of typing paper, from Whitcoulls sale gives you 500 sheets which have to be fed individually in to the printer.

$lots boxes (2000 pages) of fanfold, sprocketted, with horizontal and vertical perforations. That is, you can split at each page easily, and also tear off the sprocket hole margins to end up with a neat-edged page. (Last box from Wickliffe was $60).

3. Printer Head. This is where the wires come out, hit the ribbon to leave a dot each to form the letters of your words.

Gradually these wires wear down from all that hitting. Signs of this may be that the top dot on capital letters is darker than other dots, or the underline may also be darker - these 2 pins get less use perhaps, so are hitting harder than the other wires. Cure? When it becomes unbearable, buy a new printer head! The cost will vary among printers. The $1400 C.Itoh bought 3 years ago has had extremely heavy use (as much as the POIS one!), and is showing signs of wear. A new head was $400 for that machine. Most domestic printer-users would take twice as long at least to get to that stage of wear, and price will depend on brand of machine.

WHICH PRINTER?

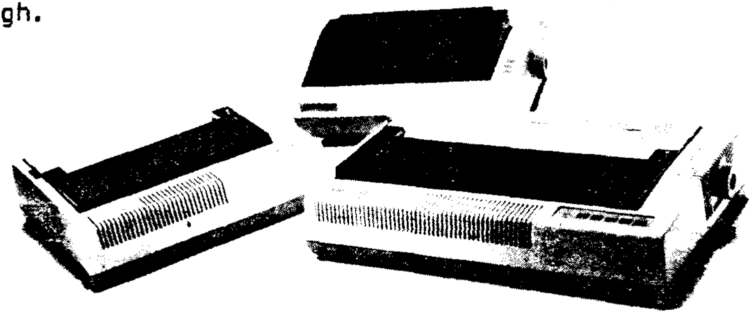
To a large extent printers are much of a muchness these days. Most of them use the same codes as Epson printers, which means that Graphics ROMs etc which include dumps for these machines will work on other printers too.

The User Group offers a choice of 3: the Solar 120, 180 and 1800.

The 120 is your basic printer, with the more economic way of feeding paper through. It prints at a nominal rate of 120 characters per second. This was fast 3 years ago. It takes up to A4-width paper.

The 180 has extra features:
a) a 3K RAM buffer, which means that the computer will have shoved out all the text and be ready to do something long before the printer has finished. Good.
b) facility to define your own characters - just like the BBC. These definitions take up some of the 3K RAM available.
c) it goes faster - nominally 180 cps

The 1800 is like the 180 except that it has a wide carriage to take the 13"(?) wide paper so beloved of computer establishments. It isn't fussy about the stripes on the paper though.



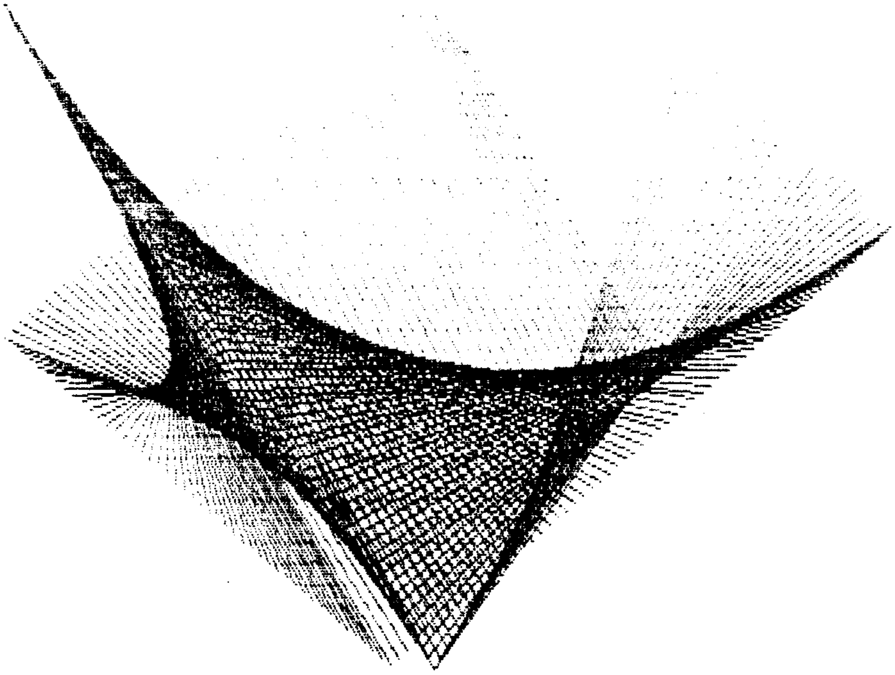
RANDOM NUMBERS

The Centaur.

Me again. This time with a Machine Code program to generate Random Numbers. Having said that I know I will receive a lot of flak because everyone knows there is no such thing as a true Random Number generator. However this R.N.G has a sufficiently long cycle (about 16 million numbers long) to be usable for all applications. The Generator can be started anywhere in the cycle depending on the initial values in locations Seed (&80) to Seed+3 (&83). The first three bytes of Seed act as a 24 bit number from which Seed+3 is extracted. This number is then juggled a bit and the result is a Pseudo-random number. The number is in the Accumulator and the X and Y registers are untouched, so if the routine is being called from M/C, Line 270 is unneccessary. Obviously this means that only numbers in the range of 0 to 255 will be produced so if bigger numbers are required, the routine may have to be called more than once.

```
   10 Seed =&80
   20 DIM E% 50
   30 FOR T=0 TO 2 STEP2
   40   P%=E%
   50   [OPT T
   60   .Rnd
   70   CLC
   80   LDA Seed
   90   ADC #&D1
  100   STA Seed
  110   LDA Seed+1
  120   ADC #70
  130   STA Seed+1
  140   LDA Seed+2
  150   ADC #180
  160   STA Seed+2
  170   LDA Seed+3
  180   ROL A
  190   EOR Seed+1
  200   ROL A
  210   EOR Seed+2
  220   ADC #0
  230   STA Seed+3
  240   LDA Seed
  250   ROL A
  260   EOR Seed+3
  270    STA Seed+4:         \ This line is only neccessary if
being
280   RTS        :      \     called from Basic
 290    ]
 300    NEXT
 310 REPEAT
 320    CALL Rnd
 330    PRINT?(Seed+4)
 340    UNTIL FALSE
```

Why use this routine when BBC Basic has its' own excellent R.N.G.? Well, two main reasons are:
1. The BBC R.N.G. cannot be called from M/C,
2. This routine is Much faster even from Basic.
The distribution of this R.N.G. is very even over large samples, and produces the correct averages etc. so even statistics students will find it useful. Well, thats it from me for another month.



PROGRAM RECOVERY AND TOKEN PRINTOUT

Steve C

When stored in memory, most interpreted computer languages use tokens to take the place of keywords. A token is a single byte (greater than or equal to &80 (128) in the BBC system) which represents the string of characters in a keyword. (Those who don't know what a keyword is will do after typing in, and RUNning, this program.)

This has a number of advantages, mainly less memory needed to store the program, as only one byte is needed for each keyword. It also makes program interpretation faster, as the interpreter only has to read one byte, rather than a whole word, of up to 8 bytes (ie RENUMBER).

Originally this program was developed to retrieve a program which had corrupted itself. From there I decided that it had another use, after it had looked up all the BASIC tokens, of printing out a list of what these are. Such a list is in the BBC UG, but not in the EUG. Beeblet Vol 3, Num 7, had a program with a similar aim, which re-arranged any remnants of a program that it could find. My program however, does not actually change memory, just reads it, and prints out any program lines it can find. Line numbers are not included, as I consider them superfluous to programming. The option is provided to SPOOL output into a file, which can be accessed with *EXEC <filename> to create a new version of the program. Such a method creates a "clean" copy of the program, as my experience with messing around in memory is that it often currupts more than it recovers. (That was why I needed this program in the first place!) Experimentation with this program, without a filename, will allow the user to find out where most of the program is, before SPOOLing it.

Having typed it in, test this program on itself!

```
 >PRINT ~PAGE : REM find the start of the program in HEX
 page
 >RUN
 start address ?<page>
 stop  address ?<page + &700) (or so)
 filename     ?<>
```

This should produce a listing of the program on screen, with no page numbers.

* * *

BASIC lines are made up as follows:-
N1 N2 LL (TO (DD DD)) &OD

N1 N2 = line Number (whats that for?)) ignored by this
LL = Length of Line) program
TO = token) these can be repeated for as many statements as there are
DD = data) on the line, and as much data as needed in each statement.
&OD = carriage return - terminates each line.

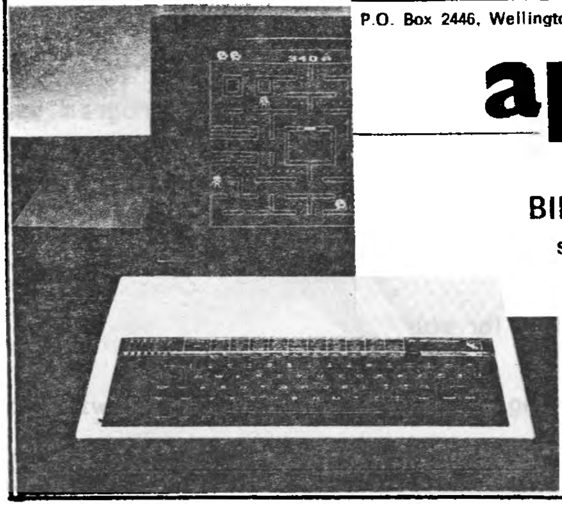
Variables
tokens$ - array holding all tokens
input$ - needed to convert input values into HEX, ends up containing <filename>
width% - number of columns required across the page in PROCprint_tokens
string$, length$ - explained in header comments in FNfield
S%, P% - start and stop addresses while recovering a program
I% - memory counter in a number of places
J% - another counter

AUTHORISED BBC SERVICE AGENTS...

APPLIED ENGINEERING & TECHNOLOGY LTD.
P.O. Box 2446, Wellington, New Zealand — Telephone (04) 850-80

aptec

BILL GATRELL
SERVICE MANAGER



```
 10 REM Program recover
 20 REM and token printout.
 30 REM by Steve Cosgrove
 40 REM 30 Sept, 1985
 50
 60 PROCread_tokens
 70 PROCprint_tokens
 80 END
 90
100 DEF PROCread_tokens
110
120 REM Designed on BASIC 2, this
130 REM probably works on either one.
140
150 REM The tokens are stored in the
160 REM ROM in alphabetical order.
170
180 DIM tokens$(&80), D% &0A
190
200 I% = &8070
210 REPEAT
220   I% = I%+1
230   UNTIL ?I% = ASC("A")
240 REM using this method to find
250 REM the first token allows for
260 REM a slight variation between
270 REM ROM versions.
280
290 REPEAT
300   $D% = ""
310   REPEAT
320     $D% = $D% + CHR$(?I%)
330     I% = I%+1
340     UNTIL ?I% >= &80
350   tokens$(?I% - &80) = $D%
360   I% = I%+2
370   UNTIL I% >= &836C
380 ENDPROC
390
400 DEF PROCrecover
401
402 REM an example of use of tokens
403 REM tries to make a BASIC progarm
404 REM out of everything between the
405 REM addresses given
```

```
410
420 INPUT "start address ",input$
430 S% = EVAL( "&"+input$ )
435
440 INPUT "stop  address ",input$
450 P% = EVAL( "&"+input$ )
455
460 INPUT "filename      ",input$
465 REM leave blank if no recording
466 REM of the program is needed
470
480 IF input$ > "" THEN OSCLI "spool
    "+input$ : PRINT "AUTO"
490 PRINT "AUTO"
500 FOR I% = S% TO P%
510
520   IF ?I% = &0D THEN PRINT : I%=I%
      +4 : REM miss out the line numb
      er
530
540   IF (?I% >= &20) AND (?I% < 127)
       THEN VDU ?I%
550   REM if its printable, print it
560
570   IF ?I% > &80 THEN PRINT tokens$
      (?I%-&80); : REM print the toke
      n
580
590   NEXT
600 PRINT
610 IF input$ > "" THEN OSCLI "spool"
620 ENDPROC
630
640 DEF PROCprint_tokens
650 width% = 2
660 FOR I% = 0 TO &7F STEP width%
670   FOR J% = I% TO (I% + width% - 1
        )
680     PRINT ; FNfield(STR$ ~(J%+&80
        ),2) ; FNfield(tokens$(J%),10
        );
690     IF J% < (I% + width% - 1) THE
        N PRINT FNfield("",9);
700     NEXT
710   PRINT
720   NEXT
730 ENDPROC
740
```

USER GROUP MEETINGS

AUCKLAND: -meets 2nd Wednesday of the month, 7.30pm, Conference Rooms, Auckland Secondary Teachers College,60 Epsom Ave, Epsom. ph Dave 770-630 x 510(bus) or Kerry 695-355.
-North Shore: meets last Wednesday of the month, 7.30pm, Salvation Army Hall, Glenfield Road, Glenfield ph Chris 419-0543 (wk)
CHRISTCHURCH: -fortnightly workshops, Monday 6.30pm at Hagley High School. ph. Michael 582-267.
DUNEDIN: -first Thursday of the month. Phone Martin McDowall 42-031
HAWKES BAY: -Alternative months Hastings & Napier. Contact Kendall Napier 435-624, Bob Taradale 446-955, Mitch Hastings 778-235.
INVERCARGILL: -Phone Evan 330-444 Invercargill.
PALMERSTON NORTH: -Phone Tom Skinner 83-009(wk)
TAUPO: -meets every second Wednesday @ 7pm. Contact David 84-215 (h), 85-124 (wk).
TAURANGA: -every second Wednesday. ph. Chris & Jane 65-076.
TIMARU: -Phone Lloyd van der Krogt Timaru 61-412.
TOKOROA: -meets first Friday of each month, Tokoroa High School, 8pm. Phone Brian 67-625 Tokoroa.
WAIKATO: -first Friday of the month 7.30pm Waikato Tech Institute. Ph. Alison or John, Morrinsville 6695.
WELLINGTON: -meets every second WEDNESDAY of the month, 7.30pm, Royal Society Lecture Theatre, Turnbull St, Thorndon.
-Workshops: second Thursday of month, other venues.
ph. Don 848-235 or Beryl 862-874 or Mike 785-437.
OTHER CENTRES: - let us know and we will publish details here.

THE NEWSLETTER:

CONTRIBUTIONS:-most welcome. Listings should be sent in on tape or disc. (Tape - 2 copies, one at 300 baud please). Please include written explanation of listing; text and graphics only material welcome too.
DEADLINES:- For a particular issue, the last day for material is the last day of the month prior to publication.
ADVERTISING:- Rates are $20 per half page (camera ready), deadline as above.

Copyright (c) BBC/ACORN COMPUTER USER GROUP OF NZ INC.
Published monthly except January and mailed to financial members.

MEMBERSHIP:

Membership of the users group is on payment of an annual subscription from April to March. For the 1985/6 year it is: $30.00, plus $5.00 for Auckland & Wellington members as a local branch levy, which entitles attendance at meetings as detailed above. Joining or renewing members get all the back issues of the current year - those joining from 1 October may have half the annual sub waived and only receive newsletters from 1 October.

THE BBC/ACORN COMPUTER USER GROUP OF NZ INC, P.O.Box 9592, WELLINGTON.

Reproduced by ROBERTS PRINTING CO. LTD, WELLINGTON.